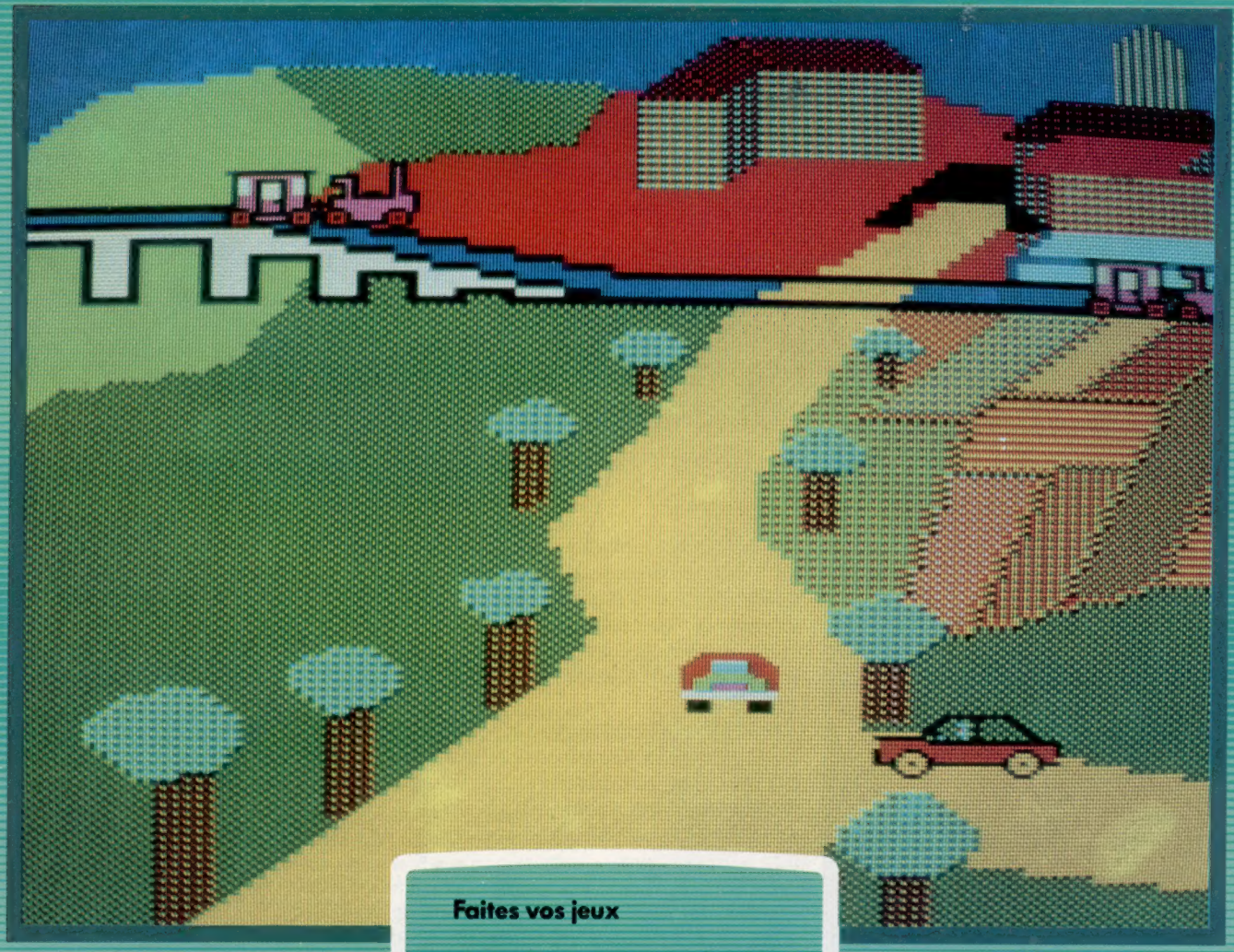
ABC INFORMATIQUE

N° 24

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Faites vos jeux

La 5e génération de micros

EXL 100 d'Exelvision

Le Basic et les autres

EDITIONS ATLAS

M 6062-24-12 F 85 FB-3,80 FS-$1.95

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume II, n° 24.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : S.M.T.-Goupil.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juin 1984. 15846. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Un risque calculé

Un jour aux courses
Les courses de chevaux représentent un intéressant terrain d'application pour les ordinateurs familiaux ; encore faut-il savoir exactement ce qu'on attend d'eux. Ainsi, il existe au moins un éleveur qui utilise un Apple II... pour établir les lignées de tous ses chevaux. (Cl. Rex Features.)

Les ordinateurs ont de nombreuses applications dans le domaine du jeu. Il existe même des programmes de pronostics sur ordinateur familial.

Le jeu est une question de probabilités et la grande majorité des joueurs perd régulièrement, parfois de façon importante. Pourquoi? Les chances favorisent le casino, le bookmaker ou l'organisme qui prend les pronostics, et non les joueurs. Pour savoir si l'ordinateur peut ou non redresser la balance, il nous faut évaluer les chances existantes.

Réduits à leur plus simple expression, tous les jeux de hasard ne sont jamais qu'un pari sur le résultat d'un événement imprévisible. On se sert généralement d'un dispositif — boule actionnée par une roulette, carte tirée d'un jeu soigneusement battu — pour obtenir l'événement aléatoire recherché. Si les paramètres — le nombre de cartes, par exemple — sont connus, la théorie des probabilités peut faire certaines prédictions. La roulette en usage dans les casinos comporte, par exemple, 37 cases, numérotées de 0 à 36. Il y a ainsi 18 numéros pairs, 18 numéros impairs, et le zéro. La probabilité pour que sorte un numéro impair est ainsi de 18 sur 37, soit un peu mieux que 48,6 %. C'est un tout petit peu moins que la probabilité de voir une pièce de monnaie retomber du côté face. Ce « tout petit peu moins », dû à l'existence du zéro, fait la différence et représente la marge de profit du casino.

C'est pourquoi les jeux de hasard, tout comme les jeux en général, sont à ce point rebelles à toute prédiction. En dépit des assurances des pronostiqueurs « illuminés », il n'est pas possible de remédier à la situation, même avec un ordinateur. On a même calculé que sur une longue période il est impossible de gagner à tout jeu de casino, sauf peut-être au black jack (ou vingt-et-un).

Les inventeurs ne se sont pas découragés pour autant. Les possesseurs d'ordinateurs peuvent désormais s'offrir des programmes de calcul garantis fiables, qui semblent même donner parfois certains résultats. Dans le cas des jeux de casino, ils se réduisent pour l'essentiel à des variations sur une vieille tactique qui consiste à doubler la mise périodiquement. Le gros inconvénient est d'exiger au départ des sommes importantes. Autre difficulté : les établissements de jeux n'acceptent pas l'usage d'ordinateurs, bien que ce ne soit pas illégal à proprement parler. L'informatique a, par contre, un rôle plus important dans les jeux qui nécessitent une certaine habileté, ou une réflexion stratégique : l'ordinateur peut alors donner au joueur la discipline nécessaire, et lui servir d'aide-mémoire. Mais la valeur d'un tel soutien tend à être inversement proportionnelle à l'adresse de celui qui joue.

Les pronostics de football présentent un problème un peu différent. Contrairement à ce qui se passe en France, on peut en Grande-Bretagne parier sur les résultats des matches, et les som-

mes mises en jeu sont énormes. Le programme le plus célèbre est le F4 Football Forecast, du professeur Frank Georges, disponible sur la plupart des ordinateurs familiaux. Il résulte de dix années d'analyses statistiques et attribue une valeur moyenne aux résultats de chaque équipe. Les résultats à long terme, les résultats à court terme et l'issue du dernier match sont mis en parallèle; la comparaison de ces éléments d'appréciation permet de prédire le résultat d'un match donné.

Il va de soi que, lorsqu'une tête de série reçoit chez elle une équipe en fin de classement, les jeux sont pratiquement faits d'avance. Pourtant le programme est capable, dans une certaine limite, d'annoncer l'issue d'un match entre formations de même niveau. Cela ne veut pas dire qu'il soit infaillible. Mais, selon les statistiques, il permet de tripler les chances de succès. Reconnaissons, toutefois, qu'il s'agit avant tout d'un jeu agréable pour l'esprit.

Les professionnels restent pourtant sceptiques. Littlewoods, l'une des plus grandes firmes britanniques de pronostics, précise qu'aucun gros gagnant n'a jamais eu recours à un ordinateur personnel. « S'il y avait un système qui marche réellement, nous le saurions » estime un porte-parole de la société. Celle-ci, d'ailleurs, procède à l'enregistrement des paris grâce à des machines automatiques, et n'utilise l'informatique que pour la tenue des comptes.

Les courses de chevaux semblent offrir plus d'espoir aux programmeurs. Il ne semble pas qu'en France on ait mis au point un logiciel qui permette de gagner au tiercé. En Grande-Bretagne, un lycéen de Darlington nommé David Stewart en a rédigé un qui aurait donné les gagnants à plusieurs reprises. Le programme a été écrit à l'origine pour le ZX-81; il est aujourd'hui disponible sur Spectrum. Les pronostics sont régulièrement diffusés sur plusieurs réseaux locaux de la B.B.C.; mais David Stewart lui-même n'a pas fait fortune...

Rien ne va plus
A la roulette, c'est le zéro qui permet au casino d'être gagnant. Il est impossible d'améliorer les chances de départ. C'est pourquoi tout programme destiné à ce jeu doit en fait se limiter à la mise au point d'un système. (Cl. David W. Hamilton.)

N'est-il pas significatif que le monde des courses ait largement rejeté l'usage de l'informatique? Le Jockey Club procède toujours manuellement au calcul des handicaps — bien que les données soient aujourd'hui stockées sur ordinateur. Dans un grand journal français spécialisé dans les courses de chevaux, on répond par une boutade : « Nous ne nous servons de l'ordinateur que pour établir la vitesse de chaque monture, en faisant intervenir l'effet du vent. Il n'existe aucun système informatisé permettant de déterminer les handicaps. En fait les ordinateurs sont tout simplement incapables de prendre en compte les résultats inattendus qui surviennent chaque jour. »

Droit au but
Plusieurs programmes pour micro-ordinateurs sont disponibles, en Grande-Bretagne, pour ceux qui désirent augmenter leurs chances sur les pronostics des matches de football. Les meilleurs d'entre eux font usage de nombreuses données relatives aux précédents résultats obtenus par les équipes. Ils peuvent parfois fournir des indications intéressantes. Mais, comme dans toute forme de « JAO » (jeu assisté par ordinateur), ils n'ont qu'une utilité très relative. Les producteurs ne s'en vantent pas...

Toutefois, l'informatique joue un rôle de plus en plus grand dans l'organisation des paris : les employés du Pari mutuel apprennent à se servir de calculatrices spécialisées, afin de calculer les sommes attribuées aux gagnants. En Grande-Bretagne, un joueur qui dispose d'un compte chez un bookmaker peut parier par téléphone; on entre son pari dans l'ordinateur, qui débite le compte de la mise correspondante, et le crédite de ses gains en cas de victoire.

Cependant, les pronostiqueurs professionnels ne croient guère aux pronostics sur ordinateur. « Personne n'a jamais gagné de cette façon, sinon nous ne serions plus là » déclarent-ils. Pourtant, une compagnie de bookmakers britanniques se livra à la simulation sur ordinateur la plus controversée de tous les temps. Les performances des précédents vainqueurs du célèbre Derby furent incorporées à un programme spécialement rédigé. Des lecteurs de journaux furent ensuite invités à prédire quels seraient les six premiers en comparant leurs carrières respectives. Mais la polémique éclata lorsqu'il fallut classer le grand cheval italien Ribop, qui n'avait jamais perdu une course. L'ordinateur le plaçait en quatrième position!

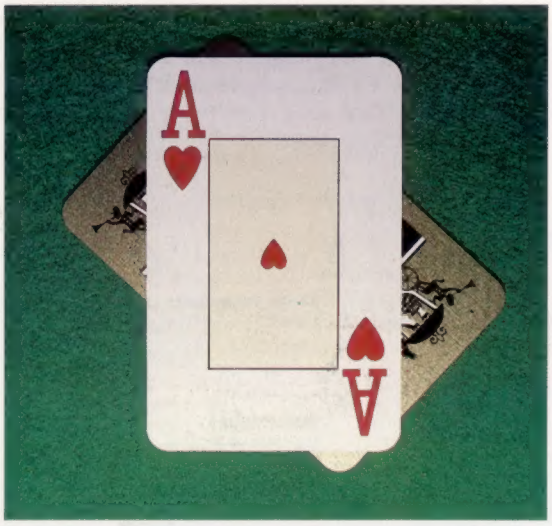
L'appareil le plus célèbre reste sans doute ERNIE (*Electronic Random Number Indicator Equipment,* « dispositif électronique de création de nombres aléatoires »). Sa tâche est de déterminer les numéros des tickets gagnants. Ce

n'est pas à proprement parler un ordinateur, puisqu'il est préprogrammé (il exécute parfaitement « le » programme pour lequel il a été construit). Il génère 200 000 nombres, qu'il inscrit sur bande magnétique. Il utilise comme base de départ les numéros des tickets, organisés en séries, qui vont du plus bas au plus haut. La bande passe ensuite sur un gros ordinateur, où les nombres sont comparés avec une liste des numéros déjà remboursés. Une fois ceux-ci éliminés, l'ordinateur peut prévenir les gagnants et leur faire parvenir les gains correspondants.

Les deux ERNIE, depuis leur mise en service, ont ainsi calculé les chiffres correspondant à 22,2 millions de prix différents, d'un montant total qui dépasse les 10 milliards de francs. Cela peut paraître impressionnant; pourtant, il faut savoir que la probabilité de gagner, au cours de chaque tirage mensuel, est de 1 sur 15 000.

Les quotidiens font également des propositions de ce genre. Qui repousserait l'idée de gagner un des millions de francs qui sont offerts? Pourtant les chances sont encore plus faibles. Les lecteurs reçoivent une carte sur laquelle est inscrit un nombre comportant douze chiffres. Une séquence de ce genre peut se décomposer en mille milliards de combinaisons possibles, allant de 000000000000 à 999999999999. Statistiquement, la chance qu'un nombre particulier puisse sortir un jour donné est un peu supérieure à 1 sur 1 000 milliards (les journaux ayant deux éditions par jour). Il est même très improbable que les 10 millions de francs doivent jamais être payés à qui que ce soit.

Un exemple concret permettra de se faire une meilleure idée. Imaginons un sac contenant 2,5 millions de boules blanches (le nombre de cartes en circulation) et 1 000 millions de boules noires, représentant l'ensemble des combinaisons possibles. Il va de soi que les possibilités de sortir une boule blanche au premier essai sont minimes. De plus, jouer un an durant ne permet pas d'augmenter les chances de façon significative. Les statisticiens estiment que la probabilité pour les organisateurs d'avoir un jour à distribuer le prix tant convoité est de 1 sur 667...



La main du destin
Le vingt-et-un ou black jack existe en de nombreuses versions informatisées; c'est l'un des rares jeux de ce genre qui permette la rédaction de programmes gagnants.
Un joueur peut accroître ses chances s'il parvient à se souvenir des cartes déjà jouées. Les casinos interdisant l'usage des ordinateurs, tous les cas révélés d'utilisation de l'ordinateur montrent que l'appareil était dissimulé (c'est ainsi qu'un joueur en avait placé un contre sa jambe, sous son pantalon), ou que des liaisons radio étaient établies avec l'extérieur.

Lancer les dés

Un jeu de hasard a d'abord besoin de nombres aléatoires, ce qui est très facile à obtenir avec un ordinateur. La plupart des versions BASIC disposent d'une fonction qui génère de tels nombres. Toutefois, ce processus n'est pas totalement aléatoire. Considérons le programme suivant :

```
10 LET A = RND
20 LET B = RND
30 LET C = RND
40 PRINT A, B, C
```

Dans les trois premières lignes, trois variables se voient attribuer des valeurs aléatoires, qui sont ensuite affichées. Les résultats pourront (ne vous attendez pas à obtenir les mêmes!) être ceux-ci :

```
.014007   .964370   .457397
```

Mais si vous faites un nouvel essai, vous vous rendrez compte que les nouveaux nombres sont les mêmes... La raison en est la suivante : lorsqu'on demande à l'ordinateur un nombre aléatoire, il en choisit un parmi une suite déjà déterminée. En général, ce nombre est choisi parmi une séquence comprise entre 0,000000 et 9,999999, et sort une fois pour chaque cycle entier — mais, bien entendu, pas dans l'ordre.

Certains BASIC procèdent différemment et ont recours à une expression entre parenthèses, qu'on appelle un *argument,* et qui prend la forme LET A = RND (X).

La fonction RANDOMIZE, dont disposent certaines variantes de BASIC, permet de faire démarrer la séquence en un point imprévisible. On peut en particulier la placer au début d'un programme qui utilise plusieurs fois la fonction RND; les nombres produits seront différents à chaque occasion.

Pour reproduire le lancer d'un dé, nous avons besoin de nombres entiers compris entre 1 et 6. Il est nécessaire d'éliminer les nombres fractionnels. On y parvient en employant la fonction INT (entier). PRINT INT(6,99) nous donnera 6, aussi sûrement que PRINT INT(6,01); tout ce qui est après la virgule est ignoré.

La fonction RND ne pouvant produire de nombre supérieur à 0,999999 (ce qui, exprimé sous forme entière, donne 0), il faut procéder à une multiplication. La formule classique est :

```
LET A = INT(6*RND) + 1
```

Nous multiplions par six parce qu'un dé a six faces. Ajouter 1 au résultat permet de s'assurer qu'il sera compris entre 1 et 6, et non entre 0 et 5.

Ligne d'assemblage

Poursuivons notre introduction au code machine par un aperçu de la variété des formes sous lesquelles un programme peut être écrit, du binaire au langage assembleur.

Une des difficultés conceptuelles rencontrées par les débutants pour écrire en code machine est la variété de ses formes. Toutes les données stockées en mémoire prennent nécessairement la forme de nombres binaires sur huit bits. Leur listage donne de longues énumérations, difficiles à déchiffrer et à retenir, sujettes à des erreurs de frappe. Aussi est-il préférable d'utiliser des nombres hexadécimaux. Ceux-ci permettent d'exprimer tout octet par un nombre à deux chiffres, et toute adresse (de 0 à 65535 en décimal), par un nombre à quatre chiffres.

Un nombre hexadécimal s'écrit précédé d'un signe $, pour le distinguer d'un nombre décimal. Le $ ne figure pas en mémoire. Lorsqu'un code opération est sur deux octets (ainsi LDA $3F80), les deux octets sont entrés dans la machine dans un ordre inverse. L'octet de poids faible est alors suivi par l'octet de poids fort. Dans l'exemple donné, les trois octets seraient AD (représentation hexadécimale du code opération LDA en langage 6502), suivi de 80 et de 3F. Ce qui est plus facile au processeur risque donc de prêter à confusion.

Un code machine est listé en hexadécimal. En outre, une adresse de début est donnée (en décimal ou en hex) pour recevoir la première valeur hex. La deuxième valeur est attribuée à l'adresse suivante, etc. Le chargement peut se faire par la commande BASIC POKE. Si l'adresse de début ou adresse initiale était $1000 (4096 en décimal), et le listage hex :

```
AD    (173 en décimal)
80    (128 en décimal)
3F    (63 en décimal)
```

le programme serait chargé par les trois instructions BASIC suivantes :

```
POKE 4096,173
POKE 4097,128
POKE 4098,63
```

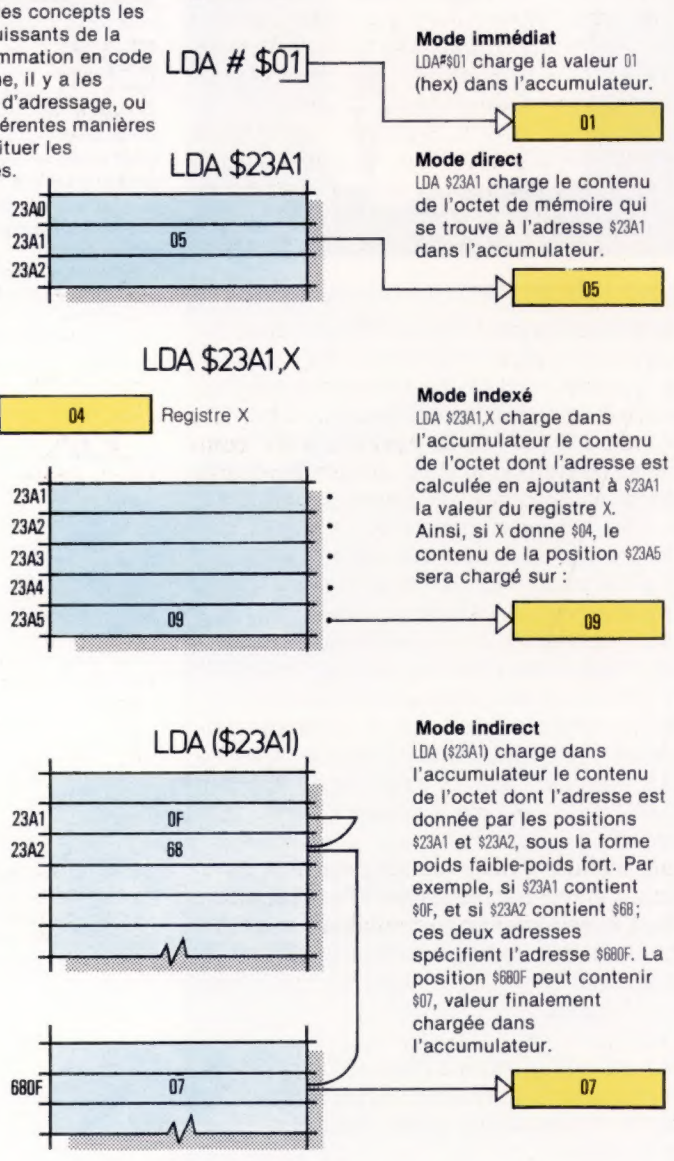
Remarquez qu'il nous faut convertir toutes les valeurs hex en décimal, pour les utiliser dans les instructions POKE. Elles seront stockées en binaire.

Pour des listages hexadécimaux importants, on utilise un court programme BASIC appelé « programme chargeur de code machine ». Il demande l'adresse initiale puis les valeurs hex. A chaque frappe, cette routine convertit en décimal et introduit la valeur par POKE sur la prochaine position mémoire disponible. Il est également possible de lire (READ) le listage hex depuis le programme par des instructions DATA.

Une fois le code machine chargé, on peut se passer du programme chargeur. Aussi est-il important de charger le code machine sur une zone mémoire qui ne sera pas effacée par le programme BASIC. Elle ne devra pas non plus être susceptible d'être neutralisée par des instructions BASIC telles que NEW.

Modes d'adressage

Parmi les concepts les plus puissants de la programmation en code machine, il y a les modes d'adressage, ou les différentes manières de restituer les données.



Écriture de code machine

Un programme en code machine peut prendre plusieurs formes. Le langage assembleur est la forme la plus répandue. Il utilise des mnémoniques pour codes opération et des étiquettes pour opérandes. Ainsi :

```
LDA WEIGHT
ADC FUEL
STA WEIGHT
```

Spécifions les adresses de ces étiquettes, par exemple :

```
FUEL = $032E
WEIGHT = $031F
```

Un ensemble utilitaire assembleur transformera cela en un listage hexadécimal par l'intermédiaire d'un lecteur de disque. Le langage pseudo-assembleur tel qu'on le montre ci-dessous est plus difficile à déchiffrer. Il peut cependant être incorporé dans un ensemble d'utilitaires appelé *spot assembler.*

```
LDA $031F
ADC $03EE
STA $031F
```

Un listage hexadécimal comprend une adresse initiale (à gauche), et une suite de deux valeurs hexadécimales sur deux chiffres figurant en mémoire. Remarquez qu'une opérande telle que $031F est stockée à l'envers (1F03), et que les codes opération ont été remplacés par leurs valeurs respectives.

```
19C4 AD 1F 03 6D EE 03 8D 1F 03
```

La plupart des ordinateurs domestiques comportent une commande BASIC pour dire à la machine de ne plus exécuter le programme BASIC, mais d'exécuter le programme code machine qui débute sur une position mémoire précise. SYS 4096 par exemple (RETURN) signifie « passer le contrôle sur le programme système commençant en 4096 ». CALL $E651 en est une autre forme et veut dire « appel de la routine code machine qui commence sur l'adresse hex E651 ».

Le sous-programme ou le programme code machine exécute alors ce programme système ou cette routine. Le résultat, selon la nature du programme, sera perceptible ou ne le sera pas. Si la syntaxe est bonne, il comporte la procédure appropriée de « fin », le contrôle revenant au BASIC. Ce qui signifie que les sous-programmes code machine peuvent être appelés en plusieurs endroits du programme BASIC à chaque fois qu'une fonction doit être exécutée très rapidement.

Une des difficultés de la programmation en code machine est que l'ordinateur ne vous adressera pas de message d'erreur, SYNTAX ERROR, lorsqu'il y a une erreur. Mais, plutôt, il se « plantera » : la machine ne réagira plus à ce que vous lui tapez. Cela n'endommage pas l'ordinateur, mais il vous faudra le réinitialiser (RESET ou éteindre-rallumer). Il vous faudra aussi retaper le programme effacé (« écrasé »). Aussi il est évident que l'on ne peut pas faire des essais en code machine comme en BASIC. Le bon déroulement du programme devra d'abord être vérifié sur le papier.

Il existe cependant un périphérique qui aide à la saisie et à la vérification du code machine : le moniteur code machine (rien à voir avec un écran moniteur). Il est parfois incorporé à la ROM, mais il est souvent sous la forme d'une cassette ou d'une cartouche. Il s'agit d'un système d'exploitation simple qui affiche à l'écran une image mémoire pour la zone choisie. Les valeurs affichées peuvent facilement être modifiées, ce qui fait du moniteur code machine le moyen le plus rapide pour saisir un listage hex. Il vous permet en outre de charger et de sauvegarder directement sur cassette les programmes en code machine, sans passer par le programme chargeur BASIC. Les plus sophistiqués des programmes utilitaires code machine (le code machine équivalant aux ensembles utilitaires) permettent d'avoir accès aux registres internes du processeur.

Les images mémoire hexadécimales, pour utiles qu'elles soient, ne sont pas faciles à lire. A moins que vous ne sachiez par cœur les équivalents hexadécimaux des divers codes opération, il est pratiquement impossible de distinguer les codes opération des opérandes. C'est pourquoi les programmes sont écrits à l'aide de mnémoniques sur trois lettres ainsi que nous l'avions décrit précédemment. La traduction en hexadécimal se fait ensuite à l'aide d'une table de codes à partir du manuel du microprocesseur.

Un moniteur code machine plus sophistiqué vous permettra de taper le programme sous la forme de mnémoniques, assurant automatiquement la conversion. Il s'agit du type *spot assembler* (assembleur instantané), qui tire son nom du fait que l'assemblage est en effet réalisé de manière instantanée.

La dernière forme possible d'assemblage est le langage assembleur. Non seulement il utilise des mnémoniques pour codes opération, mais il manie aussi des noms (ou étiquettes) à la place d'opérandes. Ainsi, si la position $07B2 comporte par exemple le nombre courant de missiles tirés dans un jeu, nous pouvons effectuer le chargement correspondant avec l'instruction :

```
LDA MISSIL
```

Au commencement du programme, il faut d'abord spécifier la position de MISSIL : $07B2. Il faut également spécifier que cette adresse doit contenir la valeur $09 (9 missiles).

Une fois le programme écrit en assembleur (code source), nous exécutons un programme utilitaire, appelé un assembleur. Il utilise le code, remplace les mnémoniques et les étiquettes par leur valeur hex, et crée ainsi une nouvelle version appelée « code objet ». Ce dernier est alors entré en mémoire d'ordinateur et exécuté. Ce traitement est semblable à une compilation, bien que dans ce cas il y ait une correspondance parfaitement exacte entre le code source et le code objet.

Le langage assembleur, langage plus évolué que le code machine, est beaucoup plus facile à l'écriture, et avec des performances égales. Cependant, les utilitaires assembleur nécessitent un lecteur de disque et ne peuvent donc pas être utilisés sur tous les ordinateurs domestiques.

Codes opération

Voici d'autres codes opération possibles avec certains micros.

JSR

Jump to Subroutine
(« passage à un sous-programme »). Fonction équivalant à l'instruction BASIC GOTO. JSR $354D exécutera le code à partir de $354D.

RTS

Return from Subroutine
(« retour d'un sous-programme »). Lorsqu'il rencontre l'instruction RTS, le processeur retourne sur la position d'appel du sous-programme (l'équivalent de RETURN en BASIC). RTS ne comporte pas d'opérande, l'adresse de retour aura été automatiquement stockée.

BMI

Branch if Minus
(« branchement si négatif »). Une des multiples formes de branchement conditionnel en code machine (en BASIC IF...THEN GOTO). Si la dernière opération a laissé un résultat négatif dans l'accumulateur, l'exécution passera à l'adresse spécifiée. BPL spécifie *Branch if Plus.*

LDX

Load X register
(« chargement du registre X »). X est un registre sur un seul octet dans le microprocesseur. Il ne peut assurer des fonctions arithmétiques comme l'accumulateur.
LDX charge une valeur dans X, et STX (STore X) la remet en mémoire.

INX

INcrement X
Ajoute 1 à la valeur de X (DEX [DEcrement X] enlève 1). En utilisant l'adressage indexé, il est possible d'appliquer ce code opération aux positions mémoire voulues.

Futurible

Les ordinateurs domestiques se sont développés de façon considérable ces cinq dernières années. Faisons le point sur les recherches en cours.

A quoi ressemblera l'ordinateur personnel des années quatre-vingt-dix? Comment fonctionnera le micro de demain? Nous allons essayer de répondre à ces questions. Pour cela, faisons le tour des divers composants et systèmes de ce que sera la machine du futur. Les idées avancées ici se fondent sur de nouvelles technologies qui font déjà leur entrée sur le marché, mais aussi sur le développement probable de ce qui nous est familier.

Une des caractéristiques principales du système que nous imaginons comme probable demain est la modularité. A partir de l'unité de base, de nombreuses extensions seront envisageables. L'utilisateur aura pratiquement choisi sa machine, une fois retenus le module graphique et le mode son présentés ici. Une chose est sûre, les changements sur le marché vont continuer à s'accélérer.

1. Affichage du clavier

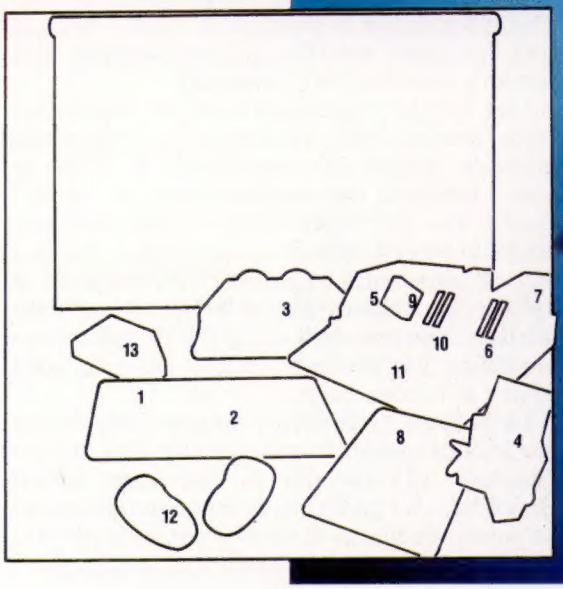
La puissance du microprocesseur 32 bits autorisera plusieurs affichages simultanés, sous diverses formes. Par exemple, l'écran principal pourra représenter la vue depuis le poste de pilotage d'un vaisseau spatial. Un écran de contrôle, devant le clavier-console de commande, donnera alors les informations de bord.

2. Clavier

Malgré l'inefficacité du clavier traditionnel, de type AZERTY, il est peu probable que l'on cherche une autre disposition des touches ou une autre conception du clavier. Ceux du type machine à écrire sont toujours très appréciés, même si les touches à électroaimants sont amenées à remplacer, ce qui est probable, les touches à ressorts.
Un nouveau système de touches, dont la frappe interrompra des rayons laser (suscitant ainsi le codage approprié), viendra vraisemblablement remplacer les touches magnétiques elles-mêmes.

3. Moniteur

Les écrans de télévision sont compatibles depuis le début des années quatre-vingt, mais leur champ est limité par les faibles ressources des tubes cathodiques. Des progrès permettent d'envisager des projections géantes. Les projecteurs de télévision devaient faire usage d'écrans incurvés, mais les derniers modèles utilisent des écrans rigoureusement plats.



4. Multiples processeurs

Outre le processeur principal sur 32 bits, il est probable que les micros des années quatre-vingt-dix pourront accueillir plusieurs processeurs annexes sous la forme de modules enfichables. Pour permettre le fonctionnement d'un périphérique ou pour le tri d'un fichier de données, le traitement des opérations sera confié à un processeur spécifique à chaque cas. Une autre solution sera l'enfichage de modules bon marché capables de remettre au niveau les micros classiques de nos années quatre-vingt.

5. Mémoire vive

Le processeur à 32 bits peut adresser jusqu'à 4 300 millions de positions mémoire — un saut vertigineux à côté de la limite de 65 536 octets des processeurs à 8 bits des débuts.

6. Communication

Alors que les antennes, pour la réception des signaux émis par satellite, seront très répandues dans les années quatre-vingt-dix, et que les réseaux du téléphone seront pour la plupart numérisés (à la place des signaux analogiques), le besoin de réguler la vitesse de transmission et la réception des signaux existera toujours.

7. Alimentation

La surcharge imposée par la multiplicité des périphériques exigera une plus grande puissance électrique. Cela supposera des circuits parfaitement réglés et une alimentation auxiliaire rechargeable pour remédier aux sautes de tension et éviter les pertes ou erreurs de données.

8. Écran portable

La technologie à écran plat, selon une matrice à cristaux liquides et connectée par infrarouges ou même par micro-ondes, s'appliquera peut-être à l'affichage texte et graphique. Si ce périphérique était également à écran tactile, il permettrait de choisir directement parmi le menu des fonctions affichées.

9. ROM sur disque compact

Celui-ci utilise un rayon laser pour lire les informations codées optiquement. Il remplacera probablement les cartouches ROM du fait de sa très grande capacité d'archivage (4 méga-octets).

10. Disques souples

D'ici à la fin de la décennie, les disques souples seront directement compétitifs par rapport aux disques Winchester, tant en vitesse qu'en densité. Ils devraient aussi diminuer en diamètre jusqu'à atteindre moins de 3 pouces.

11. Tableau frontal

Sur les premiers ordinateurs, avant l'apparition des langages évolués et des claviers, les programmes étaient entrés en binaire par l'intermédiaire du tableau frontal — ensemble de témoins lumineux et d'interrupteurs pour le contrôle des adresses, des données et des BUS de contrôle. Cette idée peut toujours être exploitable et elle enthousiasmera les passionnés d'informatique.

12. Liaison infrarouge clavier

Sur les IBM PC et Junior, la liaison clavier est déjà assurée par une radiation infrarouge au lieu d'un câble. Cette technologie pourrait assurer l'interconnexion entre tous les périphériques (dont les « souris »), supprimant les cordons torsadés. Les modèles pour gauchers et droitiers existeraient toujours, bien sûr.

13. Microprocesseur 32 bits

Les premiers micro-ordinateurs avec microprocesseurs 32 bits sont apparus en 1983, mais dépendaient de BUS de données sur seulement 16 ou même 8 bits, pour la compatibilité avec les composants mémoire et périphériques. Aussi ils ne purent tenir leurs promesses. Avec l'apparition de systèmes tels que le composant Motorola 68032, qui permet un traitement sur 32 bits et également le transfert des données sur 32 bits, les processeurs 32 bits deviendront le standard. (Cl. Tony Lodge.)

Saut de génération

Avec l'introduction de la technologie dite « intégration à très grande échelle », nous entrons dans la quatrième génération d'ordinateurs. Mais les Japonais définissent déjà la cinquième.

Les révolutions se font avec des hommes jeunes, dit-on. Le directeur du projet japonais de conception d'ordinateurs de cinquième génération semble, lui, s'être attaché à choisir des hommes nouveaux pour créer l'Institute for New Generation Computer Technology de Tōkyō. Ce projet, associant gouvernement et industrie, a retenu quarante chercheurs de moins de trente-cinq ans parmi les dix premiers groupes industriels du pays. Cet institut, créé le 14 avril 1982 et doté de plusieurs centaines de millions de francs, s'est donné pour objectif de dépasser de manière décisive d'ici à dix ans le niveau technologique actuel de l'informatique. Des sociétés telles que Fujitsu, Sharp et Toshiba participent à ce projet ambitieux devant aboutir à la cinquième génération d'ordinateurs.

Ce terme a fait prendre conscience de l'évolution de l'informatique. La première génération se caractérisait par des lampes à tube. Elles furent remplacées lors de l'invention du transistor. La deuxième génération à transistors fut dépassée par les machines à technologie LSI (*Large Scale Integration,* « intégration à grande échelle ») qui a permis de réunir de très nombreux transistors sur un seul composant. Nous sommes à la fin de cette étape, et au début de la quatrième génération. Celle-ci inaugure la technologie des composants VLSI (*Very Large Scale Integration,* « intégration à très grande échelle »), réunissant jusqu'à 10 millions de transistors sur un seul composant, contre 0,25 million couramment.

IBM consacre annuellement près de 10 milliards de francs à la recherche et au développement d'ordinateurs. Le budget japonais peut donc sembler dérisoire en comparaison, mais il a l'avantage d'être consacré à la recherche pure.

Les centres d'intérêt de la science se sont déplacés en un siècle de l'énergie à l'information. La maîtrise de l'énergie avec l'électricité a du reste permis cette évolution. Le pouvoir, jusqu'à présent, était une notion matérielle : terres, travail, capital et industries; il devient maintenant quelque chose d'abstrait, la « communication » et naturellement son contrôle.

Une société postindustrielle est en train de naître, société du traitement de l'information et de l'accès aux informations. Une telle société a besoin, selon certains, de machines douées de raisonnement logique, auxquelles on peut soumettre tout problème de données. Les activités humaines seront ainsi soumises au traitement informatique, mathématique, précis et rigoureux. La machine que les Japonais mettent au point s'appelle KIPS (*Knowledge and Information Processing System,* « système de traitement de l'information et de la connaissance »).

Les hommes savent convertir des états sensoriels en données — le jeu d'échecs s'estime à la vue, il correspond aussi à des positions de pièces. Mais lorsqu'il s'agit de prendre des décisions, cela devient délicat. En effet, la moindre décision fait intervenir un nombre considérable de données et de critères. Ainsi, pour l'exemple des échecs, bien que les règles soient simples, les champions eux-mêmes ne peuvent anticiper qu'une douzaine de coups environ. En principe pourtant, tout problème est décomposable en éléments logiques que l'on peut résoudre par des règles d'inférence. Cet ensemble de règles constitue la logique des prédicats. Cette forme de logique est applicable à toutes les situations humaines mais nous n'en avons pas conscience. Pour des situations complexes, les données sont très nombreuses. Ainsi, un médecin a besoin d'années d'expérience pour faire un bon diagnostic. La machine en cours d'étude au Japon doit comporter une banque de données sur laquelle les règles d'inférence interviendront. Le système doit pouvoir être facilement accessible à l'utilisateur, sans quoi il nécessiterait des experts spéciaux pour fonctionner. La machine devra pouvoir dialoguer dans la langue choisie par l'utilisateur; elle devra donc intégrer également des innovations dans le domaine de l'intelligence artificielle, domaine extrêmement complexe. L'objectif recouvre donc plusieurs sciences informatiques touchant le matériel, le logiciel, les interfaces, les systèmes experts et l'intelligence artificielle.

Le projet japonais doit considérer comme acquis les futurs progrès sur les composants. Or, avec l'accroissement de la densité des transistors sur un composant, les distances à parcourir par les électrons diminuent et les circuits intégrés fonctionnent plus rapidement.

Langage logique

PROLOG : abréviation pour *PROgramming LOGic,* développé au début des années soixante-dix par le groupe intelligence artificielle de l'université de Marseille, bien que son maître d'œuvre fût l'Américain Robert Kowalski, de l'Imperial College de Londres. Fondé sur certains principes de logique humaine, c'est probablement le langage des ordinateurs de cinquième génération. Il est également indiqué pour la création et l'interrogation des bases de données.

Les Japonais ont cependant compris que la vitesse de transmission de l'information n'était pas tout. Le logiciel a également beaucoup d'importance. Dans le jeu d'échecs, par exemple, il y a tellement de séquences possibles de mouvements (10^{120}) que le temps nécessaire à l'exploration de toutes les possibilités excéderait ce qui reste de vie à notre Soleil! Le projet doit aboutir à une capacité logique de 100 millions d'inférences (100 millions de règles applicables) par seconde. En d'autres termes, 100 millions d'inférences logiques par seconde.

Une autre manière d'accroître la vitesse de traitement serait de mettre le logiciel sous la forme d'un composant, au lieu de le mettre en mémoire et de le traiter par l'intermédiaire d'un composant d'usage universel. Cette disparition du logiciel, en tant qu'élément distinct du matériel, est la partie la plus intéressante du projet. Il existe déjà des mémoires « associées », qui comprennent des circuits logiques incorporés aux positions mémoire. Ces dispositifs permettent de reconnaître un élément d'information à partir de l'élément lui-même, sans passer par une adresse.

De tels progrès accéléreront la vitesse d'interaction entre les processeurs logiques et les banques de données. Inscrire le logiciel dans le matériel lui-même rappelle les premiers ordinateurs du type ENIAC. Mais les machines de la cinquième génération différeront pourtant de l'architecture de Von Neumann sur un point au moins.

Elles mettront en œuvre de nombreux processeurs travaillant simultanément (en parallèle), à la place d'une seule unité centrale. Cela suppose de grandes performances pour la synchronisation interne des opérations, mais cela supprimera les restrictions sur la vitesse inhérentes au traitement séquentiel des instructions. Le langage interne choisi pour KIPS est PROLOG, développé en France et en Grande-Bretagne, et fondé sur la logique des prédicats. KIPS aura également la faculté de communiquer dans plusieurs langues.

La reconnaissance de la parole est un autre grand but du projet, avec l'objectif d'atteindre 95 % de précision. Actuellement, la reconnaissance de la voix individuelle reste très en retard par rapport au succès évident de la synthèse vocale. La société NEC a cependant réussi à mettre au point une machine capable de reconnaître la parole, même si elle ne peut reconnaître qu'une seule voix et des mots préalablement enregistrés. L'ordinateur peut ainsi reconnaître les termes de la parole.

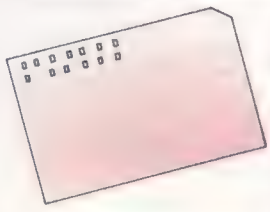
Pour ce qui concerne le langage écrit, le projet fait état d'un dictionnaire anglais-japonais et japonais-anglais de cent mille mots, et d'un programme de traduction avec un pourcentage d'exactitude de 90 %.

Le Japon a déjà connu des succès avec des programmes de recherche à long terme, comme avec le PIPS (*Pattern Information Processing System,* « système de traitement de l'image ») dans les années soixante-dix.

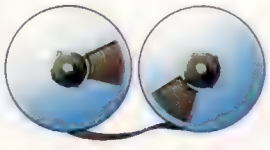
Le PIPS s'est révélé utile pour le développement des banques de données visuelles et pour les interfaces utilisateur. Le KIPS devra pouvoir extraire d'une image ses éléments significatifs afin de la reconnaître. Il existe actuellement à Tōkyō une machine qui le peut. Elle balaye à l'aide d'une caméra vidéo les couloirs du métro et donne le trafic des passagers.

La technologie de l'information a représenté en 1983 un marché de 88 milliards de dollars aux États-Unis. Ce secteur représente des perspectives importantes pour l'emploi à un moment où l'industrie connaît une crise semblable à celle qu'a connue l'agriculture il y a quelque temps dans ce pays (de 40 % à 3 % de la population active). La société tout entière va entrer dans l'ère de l'information. A la lumière de cette évolution, le projet japonais est très ambitieux. Le passage à une cinquième génération d'ordinateurs est quelque chose d'aléatoire, reposant sur d'importants progrès espérés. Mais, quel que soit le risque, cela constitue une approche positive de l'avenir.

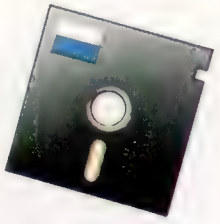
Jeu des générations



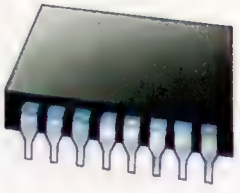
1re étape : La première génération d'ordinateurs électroniques était fondée sur la technologie des lampes à tube. Les données étaient sur cartes perforées.



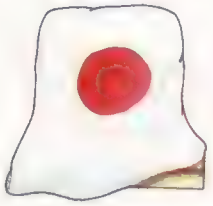
2e étape : La deuxième génération est due au développement des transistors, qui augmentèrent la capacité mémoire, bien que le stockage externe fût toujours utilisé (bandes).



3e étape : L'invention des circuits intégrés a considérablement augmenté la puissance des ordinateurs, et a permis l'apparition des micro-ordinateurs et des lecteurs de disques souples.



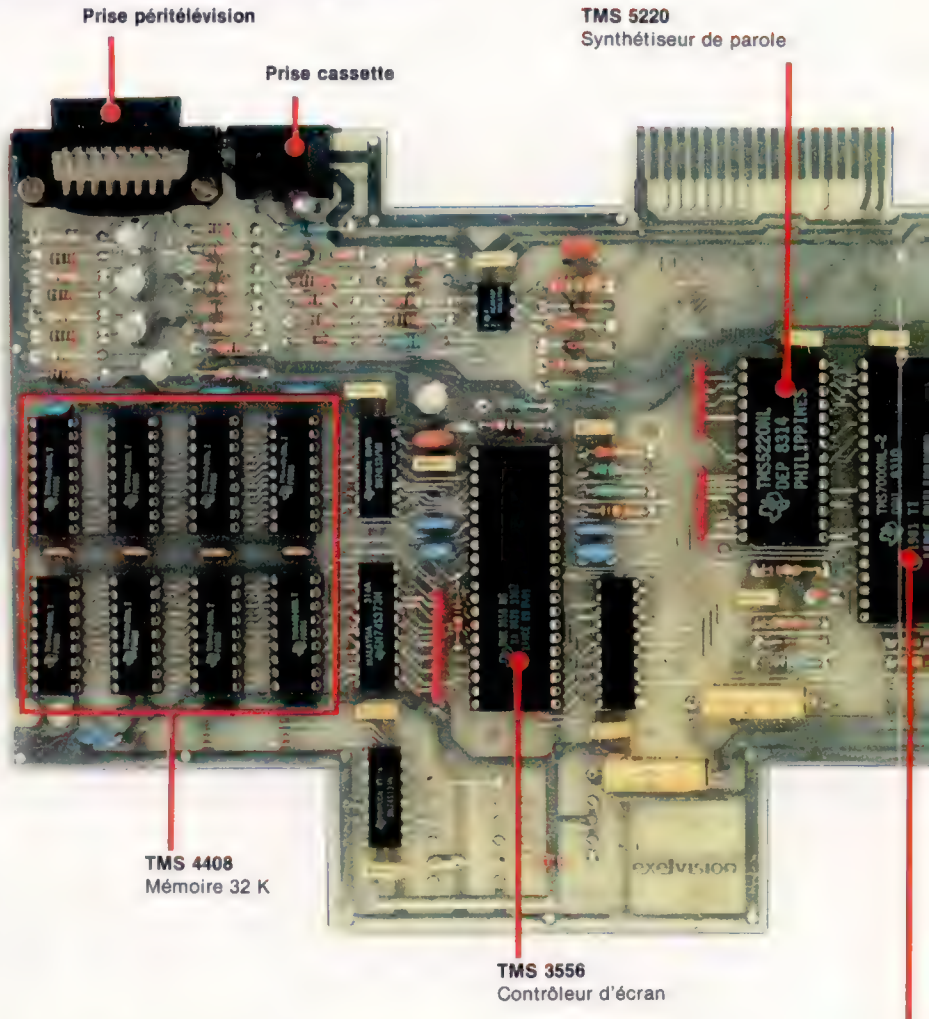
4e étape : L'intégration à très grande échelle révolutionne les composants. La mémoire RAM deviendra si grande que le stockage externe sera de peu d'importance.



5e étape : La cinquième génération d'ordinateurs se prépare au Japon. Elle concerne davantage le logiciel que le matériel. Elle repose sur l'hypothèse selon laquelle la mémoire utilisateur sera si grande que la taille du programme cessera d'être un problème. (Cl. Kevin Jones.)

EXL 100 d'Exelvision

Ce nouveau micro français se veut avant tout familial avec ses commandes à distance infrarouge, sa synthèse vocale et sa compatibilité avec les normes Vidéotex.



La micro-informatique française bouge, et nous n'avons pas fini d'en parler. Dès à présent, arrêtons-nous sur un nouveau produit arrivé avec le printemps, dénommé EXL 100. Il est fabriqué par une toute jeune société, Exelvision, installée à Sophia-Antipolis (pas loin d'IBM!), dans les Alpes-Maritimes.

Nous n'avons pu encore tester ce matériel. Sa fabrication industrielle vient juste de commencer dans une usine de la société mère d'Exelvision, la C.G.C.T. (Compagnie générale de constructions téléphoniques). Mais l'EXL 100 semble posséder, potentiellement, suffisamment d'atouts pour être d'ores et déjà décrit.

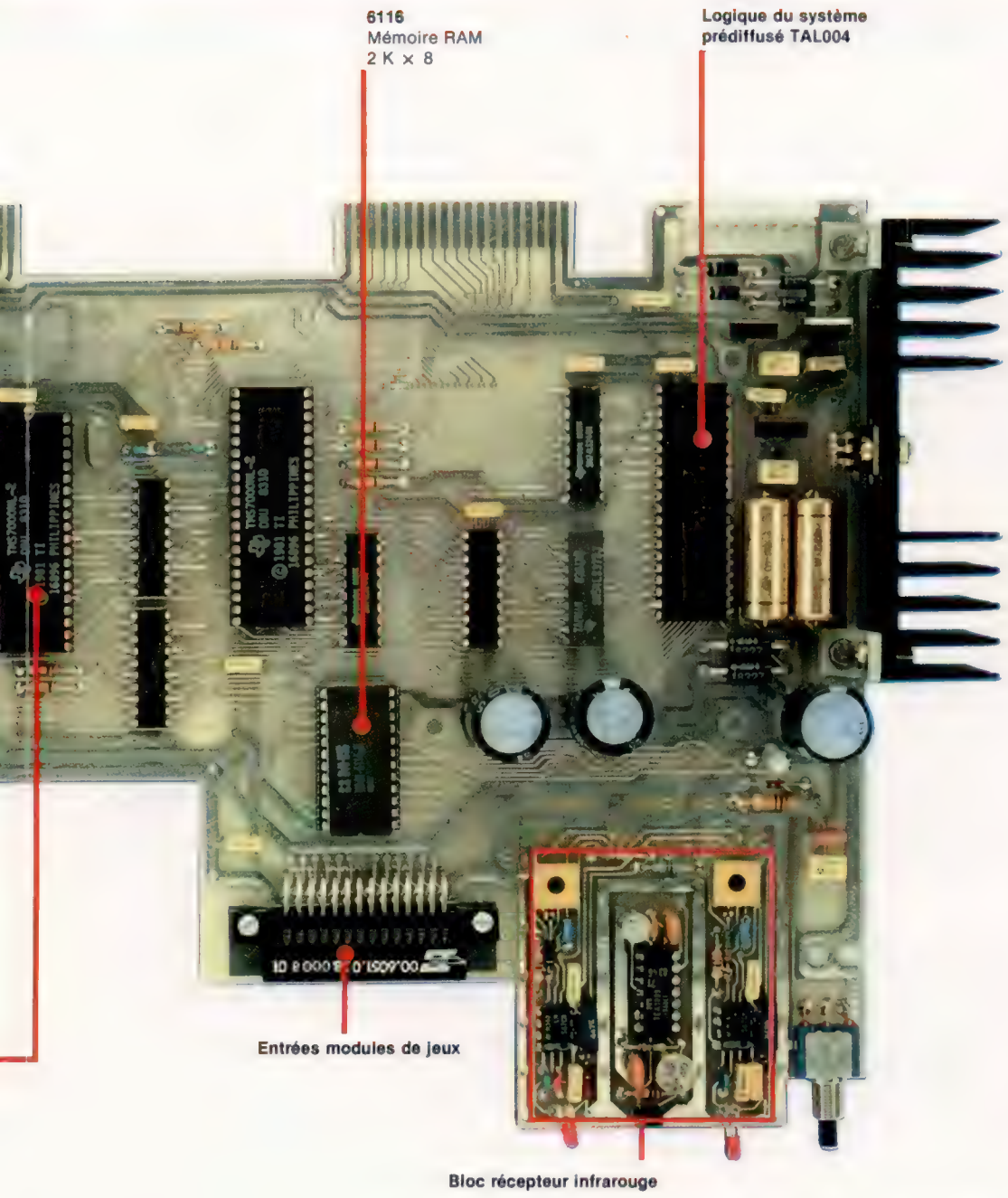
Faut-il voir là la patte de la C.G.C.T.? L'EXL 100 veut en effet se placer comme un terminal intelligent dans un vaste réseau de communications. Dans un seul et même produit, on peut trouver l'intelligence du micro-ordinateur et les possibilités de communication du Minitel. Sa vocation grand public est donc manifeste. On retrouve cette vocation sous d'autres formes.

Ainsi, le parti d'éliminer le maximum de fils de liaison entre les différents éléments du système (en profitant de la commande à distance grâce à l'infrarouge) est le gage d'une recherche de confort d'utilisation pour les grandes personnes comme pour les plus jeunes. L'ergonomie du produit va également dans le même sens, au point où l'on retrouve l'esprit des appareils domestiques qui nous sont aujourd'hui familiers comme la chaîne stéréo (les fils de raccordement en moins).

Les manettes de jeu cherchent à faciliter la tâche des enfants... et des adultes. Elles sont équipées d'un clavier de douze touches permettant l'usage des logiciels spécifiques sans passer par l'intermédiaire du clavier alphanumérique. L'EXL 100 est également équipé d'un circuit de synthèse vocale. Il est certain que pour les plus jeunes, en particulier, c'est un attrait supplémentaire.

L'EXL 100 sera mis en vente à partir de septembre 1984 dans les différents réseaux de distribution. C'est un peu la raison pour laquelle nous allons utiliser de temps en temps le futur pour donner les caractéristiques techniques de cet ordinateur familial.

L'unité centrale du système de base présente une architecture biprocesseur. Elle utilise deux microprocesseurs 8 bits et contient une mémoire morte (ROM) masquée de 6 Koctets. Les microprocesseurs choisis sont les TMS 7020 et TMS 7041 de Texas Instruments. La solution biprocesseur a été préférée, semble-t-il, pour séparer la gestion des entrées-sorties de la gestion de l'écran (l'une et l'autre sont assurées res-



EXL 100

PRIX
••

UC
TMS 7020 et TMS 7041.

HORLOGE
4,9 MHz.

MÉMOIRE
34 K RAM extensible à 290 K. 8 K ROM plus 16 K ROM BASIC, extensible à 32 K.

AFFICHAGE
250 lignes de 320 points avec huit couleurs de base mixables à l'infini.

INTERFACES
RS 232 C, cassette, mémoire CMOS RAM, infrarouge.

LANGAGE INTÉGRÉ
Exelbasic (BASIC étendu de 16 K).

LIVRÉ AVEC
Unité centrale, clavier, ROM BASIC 16 K, synthèse de la parole.

CLAVIER
61 touches mobiles dont 4 touches éditeur pleine page AZERTY accentué (majuscules et minuscules).

EXTENSIONS PRÉVUES
Imprimante standard « bus domestique » pour contrôler divers appareils domestiques, clavier musical, vidéodisque.

DOCUMENTATION
Il ne nous est pas encore possible de juger de sa qualité.

A distance
Grâce à l'infrarouge, pouvoir se mettre à bonne distance du récepteur de télévision n'est pas un vœu pieux. Exelvision a profité au maximum de cette possibilité, qui n'existe pas chez de nombreux produits concurrents. Le clavier comme les manettes de jeu permettent de rester jusqu'à 8 m de l'écran du récepteur.

pectivement par le 7041 et par le 7020). La version de base comportera 34 K de RAM (dont 32 K utilisateur) et des possibilités d'extension jusqu'à 64 K. L'extension de la mémoire morte, jusqu'à 32 K, est obtenue à l'aide de modules enfichables. Le clavier, du type AZERTY accentué, utilise une technologie que l'on retrouve sur l'IBM PC Junior : il s'agit d'une membrane en caoutchouc reproduisant de véritables touches du type de celles des machines à écrire (une matrice plastifiée indiquant les symboles de programmation se place sur le clavier, de façon à simplifier l'initiation à la programmation).

Les performances graphiques de l'EXL 100 ne semblent pas avoir été sacrifiées. Le composant retenu à cet effet est le TMS 3556, toujours de Texas Instruments, qui permet à l'ordinateur d'être compatible avec le standard Vidéotex français (40 colonnes par 25 lignes) et d'obtenir une définition graphique de haute qualité (250 lignes de 320 points, soit 80 000 pixels tous adressables et en huit différentes couleurs de base mixables à l'infini!). Arrêtons-nous enfin sur une autre innovation incluse dans l'unité de base — le circuit de synthèse vocale TMS 5220 A — qui permet, à l'intérieur du programme d'utilisation, de stocker des messages parlés. Le composant « vocal » utilise la technologie de codage par prédiction linéaire. La voix synthétique peut être ainsi compressée à un niveau de données inférieur à 1 000 bits par seconde.

Il faudra attendre encore quelque temps pour connaître toutes les ressources de ce nouvel ordinateur familial. Son prix, en tout cas, ne fera pas une révolution : 3 000 francs environ dans sa configuration de base.

Le son idéal

Le basic du Commodore 64 n'est pas à la hauteur des possibilités sonores de cet ordinateur.

Les capacités sonores du Commodore 64 figurent parmi les meilleures du marché. Elles sont dues à un composant spécial, le SID *(Sound Interface Device)*. Ce dernier présente des caractéristiques semblables aux synthétiseurs monophoniques : trois oscillateurs sur huit octaves (de 0 à 3 900 Hz en 65 536 paliers), le contrôle principal du volume, quatre formes d'ondes pour chaque oscillateur (en triangle, en dents de scie, pulsions variables, interférences), synchronisation des oscillateurs, et générateurs de niveaux sonores permettant un contrôle ADSR pour chaque oscillateur. D'autres caractéristiques doivent être retenues : modulation en anneau; filtre programmable basses et hautes fréquences; filtre programmable qui élimine une petite bande de fréquences; résonance variable; filtrage des niveaux sonores; deux interfaces de potentiomètres lecture analogique-lecture digitale pouvant être utilisés pour contrôler les caractéristiques du SID; une entrée son externe permettant l'interconnexion de plusieurs composants SID. D'autres signaux sonores peuvent être reçus en entrée, filtrés et mixés avec les sorties standard SID.

Nous ne pouvons exposer toutes ces fonctions (de bons livres existent à leur sujet), mais nous pouvons expliquer à quoi elles se rapportent. En premier lieu, la synchronisation d'oscillateurs signifie l'harmonisation de deux signaux (ici deux voix spécifiées) en une seule tonalité.

Une modulation consiste en une modification d'un signal par un autre, en changeant soit la fréquence, soit l'amplitude (volume). La modulation en anneau est une modulation d'amplitude d'une voix sur l'autre. Le résultat donne une sonorité suraiguë semblable à celle d'une cloche ou de batteries métalliques. De tels sons possèdent une dominante « inharmonieuse ».

Les filtres permettent d'éliminer d'un signal certaines gammes de fréquences. Leur nom, sur le Commodore 64, indique en général leur effet. Les filtres basses fréquences éliminent les fréquences supérieures à une certaine valeur. Les filtres à bande de fréquences suppriment les fréquences situées au-dessus d'une certaine valeur et au-dessous d'une autre. D'autres filtres éliminent au contraire toute une bande de fréquences. Les filtres hautes fréquences éliminent les valeurs inférieures à la valeur de référence (« plancher »). La résonance variable peut être

Guide d'ondes

Les figures graphiques (des missiles que dirige l'utilisateur) sont un point fort des micros Atari.

Figures missiles-utilisateur

Les « missiles-utilisateur » d'Atari sont semblables aux figures graphiques « Sprites-Lutins » du Commodore 64 et du Sord M5. Ils représentent une des preuves majeures des grandes capacités graphiques de l'Atari. Ils permettent de définir et de manœuvrer jusqu'à huit figures graphiques haute résolution. Ces dernières interviennent librement sur le fond de l'affichage et peuvent apparaître sur plusieurs plans, à l'avant ou à l'arrière d'autres formes dessinées sur l'écran. Les effets visuels se présentent ainsi en trois dimensions. Les missiles-utilisateur peuvent se déplacer graduellement à très haute vitesse; ils conviennent parfaitement aux jeux d'arcades (Atari est déjà très présent sur ce marché). Ils peuvent également servir à créer des dessins d'écran plus coloriés qu'avec les modes graphiques habituels. En effet, les missiles graphiques peuvent être coloriés indépendamment les uns des autres et du fond. L'intérêt essentiel de ces figures vient des composants électroniques utilisés. Des registres spécifiques contrôlent le mouvement, la couleur et l'affichage des missiles.

Le programmeur doit simplement placer des valeurs dans les registres appropriés, et la machine se charge du reste. Un nombre est attribué à un registre par l'intermédiaire de l'instruction POKE. La vitesse d'exécution est celle du traitement de code machine, bien supérieure à celle du BASIC.

Voyons maintenant la création des figures-missiles et les registres concernés. Une figure est façonnée à partir d'une bande verticale de 8 pixels de large sur 128 ou 256 pixels de haut. Chaque ligne de cette colonne est représentée sur un seul octet en mémoire. En passant des ordres POKE selon des codes binaires appropriés, on crée une figure, de manière très semblable à la création de caractères. Il est possible de défi-

```
10 SID=54272
20 POKESID+23,0
30 POKESID+24,15
40 POKESID+5,40
50 POKESID+ 6,201
60 FOR N=I TO 5
70 READ FH,FL,D
80 POKESID+1,FH:
   POKESID,FL:
   REM*JOUER
   NOTE*
90 POKESID+4,33
100 FOR I=1 TO 300*
   D:NEXT I
110 POKESID+4,32
120 FOR I=1 TO 100:
   NEXT I
130 NEXT N
140 FOR I=1 TO 2000:
   NEXT I
150 POKESID+24,0
160 REM**FH FL D**
170 DATA 57,172,1
180 DATA 64,188,1
190 DATA 51,97,1
200 DATA 25,177,1
210 DATA 38,126,2
220 END
```

appliquée aux filtres précédents pour accentuer les fréquences autour de leur point de rupture. Le filtrage des niveaux sonores est un cas à part : les valeurs sous forme de chiffres ADSR pour « Envelope » 3 (niveau sonore) peuvent être lues par le SID et appliquées à un signal de telle sorte que la structure harmonique change pendant une note. C'est une sorte de filtre variable.

Ces caractéristiques très évoluées vous permettent d'élaborer des sons très complexes sous la forme d'effets sonores et de timbres comparables à des instruments de musique. Le problème avec le SID est que le CM BASIC V2, fourni avec le Commodore 64, ne lui est pas approprié. En effet, ce langage ne prévoit aucune commande pour le son. Le contrôle s'exerce par PEEK depuis les vingt-neuf registres de commande du SID, et par POKE vers ces derniers. Il faut donc écrire beaucoup de code même pour générer des sons simples. Le BASIC est donc peu rapide comparé aux performances du SID. Une description exhaustive de ce composant dépasse le cadre de cet exposé, mais il est possible de l'utiliser simplement comme dans le programme donné ici, pour jouer des notes de tonalité agréable.

Bien que le programme ait 22 lignes, il ne fait que jouer cinq notes d'une petite mélodie sur un seul oscillateur. La ligne 20 déconnecte le filtre, la 30 met le volume principal à son maximum, enfin les lignes 40 et 50 spécifient un niveau sonore ou timbre semblable au piano (Enveloppe). La ligne 80 donne la fréquence de la note ; 90 et 100 commencent et terminent respectivement le cycle ADSR et retiennent une forme d'onde du type « dents de scie » pour la voix 1. La synchronisation est assurée par des boucles FOR...NEXT aux lignes 100, 120 et 140.

La programmation de sons en BASIC sur le Commodore 64 est éprouvante, tant pour maîtriser le code approprié que pour l'écrire. De plus, elle est frustrante dans la mesure où il faut procéder par essais. Si vous désirez utiliser des méthodes plus simples, vous pouvez avantageusement acquérir un « éditeur de sons ». Ils sont habituellement écrits en code machine et sont à même de tirer un meilleur profit des remarquables ressources du Commodore 64.

nir ainsi jusqu'à quatre figures prenant chacune 256 ou 128 octets de mémoire.

Chacune des quatre figures comporte le dessin d'un missile sur deux octets de large. La création de figures et de missiles suppose de placer (par POKE), dans une zone mémoire particulière, la configuration de bits qui définit leur forme. La partie de la mémoire vive choisie peut être sélectionnée par le programmeur, mais il faut l'indiquer à l'ordinateur en la faisant précéder d'un pointeur.

Si le programmeur choisit une résolution verticale sur un pixel seulement, il faudra deux fois plus de mémoire. Le programme suivant façonne la figure 0 selon une résolution verticale sur deux pixels, pour représenter un vaisseau spatial :

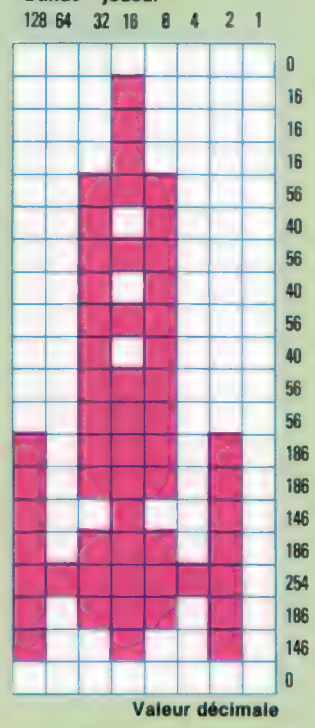
```
10 REM ***DÉFINIR UNE FIGURE***
20 P=PEEK(106)-8:REM MET P A 2 K IMMÉDIATEMENT
   DESSOUS LE DESSUS DE LA REM
30 POKE 54279,P:REM MET LE POINTEUR SUR LA ZONE
   FIGURE
40 BASE=256*P:REM DONNE L'ADRESSE DE LA VALEUR
   INFÉRIEURE DE LA ZONE FIGURE
50 FOR I=BASE+512 TO BASE+640
60 POKE I,0:REM VIDER LA ZONE DE LA FIGURE 0
70 NEXT I
80 FOR I=BASE+512+50 TO BASE+530+50
90 READ A:POKE I,A:REM DÉFINITION DESSIN
100 NEXT I
110 DATA 16,16,16,56,40,56,40,56,40
120 DATA 56,56,186,186,146,186,254,186,146
```

Chaque figure dépend de plusieurs registres. Ces derniers contrôlent la couleur, le positionnement horizontal et la taille de la figure. Le registre pour la taille permet d'agrandir une figure d'un facteur 2 ou 4. D'autres registres contrôlent la priorité à accorder au fond ou à la figure. Les missiles prennent la couleur de la figure dont ils sont issus, mais leur taille peut changer librement. Pour les applications de jeux, un ensemble spécial de registres détecte les collisions à l'écran entre les différents missiles, figures et autres éléments du décor. Cependant, il n'y a pas de registres pour la position verticale des missiles ou des figures. Le déplacement vertical s'obtient en changeant le contenu de chaque position de la configuration de bits d'une figure jusqu'à la partie mémoire réservée à cet effet pour le joueur. C'est assez rapide en assembleur, mais relativement lent en BASIC. Aussi est-il conseillé de concevoir des caractères se déplaçant très peu verticalement.

« Figure et missile » accroît considérablement les ressources graphiques d'Atari, même si elles sont moins souples que celles de « Figure et lutin » du Commodore 64. La suite du programme, donné plus haut, apporte la couleur et le déplacement de gauche à droite au vaisseau spatial :

```
130 POKE 559,46:REM PERMET L'AFFICHAGE DE LA LIGNE
    DE LA FIGURE 2
140 POKE 53277,3:REM PERMET L'AFFICHAGE DE LA FIGURE
150 POKE 704,88:REM COULEUR ROSE POUR LA FIGURE 0
160 GRAPHICS 0
170 SETCOLOR 2,8,2:REM COLORE LE FOND EN BLEU NOIR
180 FOR I=0 TO 320
190 POKE 53248,I: REM DONNE LA POSITION HORIZONTALE
200 NEXT I
210 END
```

Figure « Rocket »
Avant de définir une figure, il faut la dessiner, et calculer les valeurs décimales pour chaque ligne de pixels.
(Cl. Liz Dixon.)



Laboratoire de langue

Pour conclure notre cours, considérons d'un œil critique le basic et abordons d'autres langages.

Pour clore ce cours de programmation BASIC, nous allons déterminer quelques-unes des forces et des faiblesses du BASIC comparé à d'autres langages de programmation.

Le BASIC est un rejeton du FORTRAN, l'un des premiers langages de programmation. Contrairement à la plupart des autres langages, le BASIC est interprété; cela signifie qu'un programme spécial interprète le code ligne par ligne et convertit les instructions BASIC en langage machine. Voici ce qui arriverait à un petit programme BASIC tel que celui-ci :

```
CLS
20 PRINT «TAPER UN NOMBRE
30 INPUT X
40 PRINT «TAPER UN AUTRE NOMBRE
50 INPUT Y
60 PRINT «LE PRODUIT DES DEUX NOMBRES
70 PRINT X*Y
PRINT
90 PRINT «VOULEZ VOUS RECOMMENCER ?
100 PRINT «TAPEZ "O" SI OU
110 PRINT «OU "N" POUR ARRÊTER
120 FOR X=1 TO 1
130 LET A$=INKEY$
    A$<>«O» AND A$<>«N» THEN X
    A$=«O» THEN GOTO 1
```

En rencontrant la ligne 10, l'interpréteur BASIC trouverait le code machine nécessaire pour effacer l'écran; à la ligne 20, les instructions en langage machine pour afficher le message TAPER UN NOMBRE à l'écran. A la ligne 30, il établirait l'espace de mémoire nécessaire pour stocker un nombre réel, attendrait l'entrée à partir du clavier, puis convertirait le nombre tapé en binaire et le stockerait dans l'espace alloué à la variable X. Tout cela serait répété pour les lignes 40 à 60. Si l'utilisateur voulait recommencer, l'interpréteur se rebrancherait à la ligne 10 et réitérerait à nouveau tous les calculs.

La plupart des autres langages sont « compilés ». Cela signifie que, après qu'on a écrit le programme, celui-ci est traité par un « compilateur » avant de pouvoir tourner. Le compilateur est un programme distinct qui parcourt le « code source » (programme original) et produit une seconde version de celui-ci en langage machine. Lorsqu'on fait tourner le programme compilé, il a des chances de fonctionner beaucoup plus vite qu'un programme interprété, parce que toutes les traductions en langage machine ont déjà été effectuées.

Si les programmes compilés sont tellement plus rapides que les programmes interprétés, pourquoi tous les langages de programmation n'utilisent-ils pas de compilateurs? Les programmes interprétés, tels que BASIC, offrent un certain nombre d'avantages. La plupart proviennent du fait que c'est un langage interactif : on peut corriger les erreurs « au clavier » tout en développant le programme. Le BASIC autorise, par exemple, que l'on insère la commande STOP en n'importe quel endroit du programme. Lorsque l'interpréteur rencontre un STOP, il arrête d'interpréter le programme et permet d'émettre des « commandes » à partir du clavier.

Les commandes et les instructions peuvent être directement exécutées par l'interpréteur lorsque le programme ne tourne pas. Le BASIC en comprend un grand nombre, et elles sont inestimables pour corriger les erreurs. Après avoir exécuté un programme en BASIC (c'est-à-dire lorsque l'interpréteur a rencontré l'instruction END) ou quand l'interpréteur rencontre un STOP, il est possible d'afficher les valeurs de toutes les variables. Essayez avec le programme de carnet d'adresses. Faites tourner le programme et tapez 9 pour sortir du programme. S'il fonctionne bien jusqu'à la fin sans afficher aucun message d'erreur, il devrait se terminer par le signe BASIC (en général OK, > ou *). Ensuite tapez PRINT RMOD <R>. L'interpréteur devrait afficher un 0 à l'écran (à condition que vous n'ayez ajouté aucun enregistrement!). Puis essayez PRINT TAILLE <R>. L'interpréteur affichera un nombre à l'écran, supérieur d'une unité au nombre d'enregistrements qui se trouvent dans le fichier de données.

Les avantages du basic

On dit souvent que le BASIC est le langage idéal pour le programmeur inexpérimenté, car il permet de corriger les erreurs au clavier. Il a un autre grand avantage : il est relativement facile à apprendre. Par exemple, dans le cours de programmation BASIC, nous avons étudié tous ses aspects fondamentaux et un grand nombre de ses particularités en quatre-vingt-six pages seulement. Les erreurs de syntaxe telles que 40 PRNT A(12) feront généralement apparaître des messages d'erreur faciles à comprendre quand le programme sera exécuté, comme SYNTAX ERROR IN 40. Il suffira en général de jeter un coup d'œil à la ligne concernée pour voir où se trouve l'erreur, et pour rectifier, de taper EDIT 40 <R> (suivi de quelques instructions d'éditeur) ou de retaper la

ligne correctement. Chaque fois que l'interpréteur BASIC rencontre une erreur de syntaxe ou de logique, il arrête l'exécution du programme et liste l'erreur. Corriger les erreurs est aussi simple que d'essayer une nouvelle ligne à la place de la ligne fausse et de taper RUN <R> à nouveau.

Les désavantages du basic

Le BASIC n'a pas que des avantages. Puisqu'il est interprété (bien qu'il existe aussi quelques versions compilées de BASIC), il est exécuté très lentement. Si la vitesse n'est pas primordiale (comme dans un programme pour calculer votre compte bancaire), la lenteur du BASIC interprété n'aura aucune conséquence. En revanche, dans les cas où la vitesse est essentielle (comme dans les programmes d'animation d'écran utilisant l'affichage graphique, ou pour une « horloge » mesurant le temps de réaction dans une expérience de laboratoire), il a de grandes chances d'être trop lent.

Si vous avez besoin de vitesse dans vos programmes, deux voies sont à suivre : programmer soit en langage machine, soit en assembleur — tâche difficile et qui demande beaucoup de temps —, ou bien programmer dans un langage compilé tel que le PASCAL ou le FORTH. Les langages compilés ne sont pas difficiles à apprendre. Mais le code source (programme original) contenant presque à coup sûr des erreurs, le compilateur les trouvera lorsqu'il essaiera de compiler le programme. Celles-ci seront difficiles à rectifier si on les compare avec les erreurs en BASIC. Après avoir corrigé le code source, le programme devra être entièrement recompilé. La plupart des compilateurs font deux ou trois « passages » du code source, et chaque passage peut fournir des messages d'erreurs. Autant de corrections à effectuer avant de pouvoir recompiler le programme.

Il faut bien plus de temps pour obtenir un programme compilé correct que pour réaliser un programme qui tourne en BASIC interprété. Par ailleurs, le BASIC peut écarter le programmeur débutant du droit chemin, en lui permettant d'utiliser de mauvaises techniques de programmation que des langages très structurés comme le PASCAL rejetteraient. Le BASIC autorise le programmeur à écrire des programmes très peu soignés (abondance de GOTO), et ces mauvaises habitudes rendent plus difficile la transition vers des langages plus avancés.

Qu'est-ce qui vient après le basic?

Le BASIC est un langage flexible peu difficile à apprendre. Il offre de nombreuses possibilités, mais il est lent et ne permet pas de profiter pleinement des capacités d'un ordinateur domestique. Par contre, des langages plus modernes tels que le PASCAL et le FORTH dépassent les limites du BASIC.

Le PASCAL a été conçu comme un langage d'enseignement, particulièrement destiné à encourager le développement de programmes bien construits, « structurés ». Le PASCAL étant un langage compilé, cela signifie que les utilisateurs rencontrent de nombreuses erreurs relevées par le compilateur (après avoir écrit le code source et avant que le « code objet » ne puisse être exécuté). Les programmeurs débutant en PASCAL ont aussi tendance à trouver que les contraintes du langage, comme la nécessité de déclarer toutes les variables au début du programme (et de préciser leur type — réelles, entières, etc.), font obstacle à une programmation libre et souple.

En outre, le PASCAL exige de la part du programmeur une réflexion sur la logique du programme avant de l'écrire. Les programmes en PASCAL peuvent donner lieu à toute une quantité d'erreurs de syntaxe dans le code source.

Le FORTH est récemment devenu un langage de programmation très populaire, susceptible de rivaliser avec le BASIC sur les micro-ordinateurs domestiques. Bien que le FORTH ne soit pas aussi difficile à apprendre qu'un assembleur ou un langage machine, il faut dire qu'il est bien moins « intuitif » que le BASIC ou le PASCAL.

Quoique le FORTH soit un langage de haut niveau, il est presque aussi rapide que le langage machine, grâce à son mode unique de fonctionnement. Tandis que des langages comme le BASIC ont un nombre fixé d'instructions et de commandes, les utilisateurs du FORTH peuvent définir leur propre vocabulaire.

Le mot clé PRINT en BASIC implique que tout caractère qui le suit, encadré par des guillemets, sera affiché à l'écran. Le programmeur n'y pourra rien changer. En FORTH, on pourra définir PRINT pour produire, par exemple, un listage à l'écran des équivalents hexadécimaux des codes ASCII, affichés en colonne verticale, des caractères d'une chaîne.

Le FORTH donne au programmeur la possibilité de définir n'importe quel mot comme il l'entend et de produire les résultats voulus chaque fois qu'il en a besoin. Non seulement le FORTH est extrêmement flexible, mais il produit aussi des programmes qui peuvent être compilés dans le code objet. Ceux-ci sont presque aussi compacts et rapides que des programmes en langage machine.

Bien qu'il y ait beaucoup de langages de programmation disponibles, la plupart des amateurs qui veulent dépasser le BASIC choisissent le PASCAL, le FORTH ou l'assembleur. En résumé, voici les avantages et désavantages de chacun d'eux :

BASIC
Facile à apprendre.
Facile à mémoriser.
Erreurs faciles à corriger.
Lent dans son exécution.
Utilise beaucoup de mémoire.
N'encourage pas la programmation structurée.

Langage d'assemblage
Pas très facile à apprendre.
Pas très facile à mémoriser.

Erreurs difficiles à corriger.
Très rapide dans son exécution.
Donne un contrôle complet sur le microprocesseur.

PASCAL
Assez facile à apprendre.
Assez facile à mémoriser.
Correction des erreurs plus difficile qu'en BASIC.
Encourage de meilleures techniques de programmation.
Exécution plus rapide que le BASIC mais plus lente que l'assembleur.
Doit être compilé, ce qui prend du temps; une fois correctement compilé, va presque aussi vite que l'assembleur.
Donne un bon contrôle sur le microprocesseur, mais moins que l'assembleur; manipulation des chaînes moins facile qu'en BASIC.

FORTH
Pas très facile à apprendre; plus facile pour vrais débutants, moins pour qui connaît le BASIC.
Assez facile à mémoriser.
Correction des erreurs très facile en mode interpréteur.
Peut être compilé; presque aussi rapide que l'assembleur.
Donne un contrôle complet sur le microprocesseur.
Très économique en mémoire.
Plus facile à apprendre que le langage d'assemblage, quoique moins « intuitif » que le BASIC.

Variantes de basic

LYNX 96

```
1 REM *CRÉE LE FICHIER DE DONNÉES*
2 DIM N$(30)
3 LET N$=« @PREMIER»
4 DIM F$(15)
5 LET F$=«FACTICE»
6 LET Z=2
7 EXTBACK 1
8 EXTSTORE 1,Z,N$,N$,N$
9 EXTSTORE 1,F$,F$,F$,F$
10 INPUT «INSÉREZ LA CASSETTE DE DONNÉES,
   APPUYEZ SUR "RECORD" ET TAPEZ "Y";A$
11 SSAVE 1,«ADBKDAT»
12 PRINT «ARRÊTEZ LA CASSETTE ET
   REMBOBINEZ»
13 END
```

N.B. — Voilà le programme d'initialisation pour le Lynx 96 K; nous n'avons pas d'information concernant les fichiers sur cassettes pour les autres modèles.

Variables du programme principal
Copiez la liste du Spectrum avec les substitutions suivantes pour les variables numériques :

Remplacez	TAILLE	par Z
	RMOD	par R
	TRIE	par D
	ACT	par C
	CHOI	par H
	INF	par b
	MD	par m
	SUP	par t

et faites les changements de lignes, substitutions et suppressions suivants :

```
1100 REM S-P *CRETAB*
1110 DIM N$(30)(50)
1120 DIM M$(30)(50)
1130 DIM S$(30)(50)
1140 DIM T$(15)(50)
1150 DIM C$(15)(50)
1160 DIM R$(15)(50)
1170 DIM X$(15)(50)
1180 DIM Z$(30)
1210 LET Z=0
1220 LET R=0
1230 LET D=1
1240 LET C=0
1250 LET S$=« @PREMIER»
1260 LET Q$=«»
1300 RETURN
■
1400 REM S-P *LECFCR*
1405 PRINT «INSÉREZ LA CASSETTE DE
     DONNÉES ET APPUYEZ SUR "PLAY"»
1410 GOSUB 3100
1420 SLOAD 1, «ADBKDAT»
1430 PRINT «ARRÊTEZ LA CASSETTE»
1440 GOSUB 3100
1450 EXTBACK 1
1460 EXTFETCH 1,Z
1470 FOR K=1 TO Z-1
1480 EXTFETCH1,
     N$(K),M$(K),S$(K),T$(K),C$(K),R$(K),X$(K)
1490 NEXT K
1500 LET Q$=N$(1)
1510 RETURN
•
3120 IF KEYN<>32 THEN LET L=0
•
3780 LET A$=KEY$
■
3810 LET H=VAL(A$)
3820 IF (H<1) OR (H>9 THEN LET L=0
•
4500 REM S-P *MODNOM*
4510 REM CONVERTIT EN MAJUSCULES
4520 LET D$=UPC$(N$(Z))
•   (effacer les lignes 4530-4590)
•
4600 LET P$=«»
4601 LET A$=«»
4602 LET T=LEN(D$)
4603 LET S=0
•
4610 REM LOCALISE LE DERNIER ESPACE
•
4630 IF MID$(D$, L,1)=«» THEN LET S=L
•
4670 IF MID$(D$,L,1) > «@» THEN LET
     P$=P$+MID$(D$,L,1)
■
4710 IF MID$(D$,L,1)>«@» THEN LET
     A$=A$+MID$(D$,L,1)
```

Les lignes 5410 à 5460 doivent être réduites à des instructions simples, par exemple :

```
5410 LET U$=N$(L):LET N$(L)=N$(T):LET N$(T)=U$
```

devient :

```
5410 LET U$=N$(L)
5411 LET N$(L)=N$(T)
5412 LET N$(T)=U$
```

etc., sans autre changement.

```
■
5600 REM S-P *SAUVENR*
5605 PRINT «INSÉREZ LA CASSETTE DE
     DONNÉES ET APPUYEZ SUR "RECORD"»
5610 GOSUB 3100
5620 EXTBACK 1
5630 EXTSTORE 1,Z
5640 FOR K=1 TO Z-1
5650 EXTSTORE 1,
     N$(K),M$(K),S$(K),T$(K),C$(K),R$(K),X$(K)
5660 SSAVE 1, «ADBKDAT»
5670 PRINT «ARRÊTEZ LA CASSETTE»
5680 GOSUB 3100
5690 RETURN

•
5855 LET X=0
```

```
5860 LET m=INT((b+t)/2)
5870 IF M$(m)=U$ THEN LET X=1
5880 IF U$>M$(m) THEN LET b=m+1
•
6080 LET A$=KEY$
•
6110 IF A$="" THEN RETURN
6120 GOSUB 6200
6130 RETURN
•
6730 FOR I=1 TO 1
6735 LET I=0
6740 LET A$=KEY$
6750 IF (A$=E$) OR (A$="") THEN LET I=1
6760 NEXT I
```

Ce fragment doit être reproduit aux lignes 6880-6910, 6990-7030, 7110-7140, 7220-7250, 7640-7670.

Programme d'initialisation

Voici le programme d'initialisation pour le Dragon 32 :

```
1 REM *CRÉE LE FICHIER DE DONNÉES*
2 LET Z=2
3 LET N$="@PREMIER"
4 OPEN "O",#-1,"ADBKDAT"
5 INPUT "INSÉREZ LA CASSETTE DE DONNÉES,
  APPUYEZ SUR "RECORD" ET TAPEZ "Y";A$
6 PRINT #-1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE #-1
8 PRINT "ARRÊTEZ LA CASSETTE ET
  REMBOBINEZ"
9 STOP
```

Sur le BBC Micro, remplacez les lignes 4, 6 et 7 par :

```
4 F1=OPENOUT("ADBKDAT")
6 PRINT #F1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE #F1
```

Sur le Commodore 64 et le Vic-20, remplacez les lignes 4, 6 et 7 par :

```
4 OPEN 1,1,2,"ADBKDAT"
6 PRINT #1,Z,N$,N$,N$,N$,N$,N$,N$
7 CLOSE 1
```

Variables du programme principal

Sur le Dragon, les Commodore et le BBC Micro, copiez la liste du Spectrum (publiée en entier précédemment) avec les substitutions suivantes pour les variables numériques :

Remplacez TAILLE	par Z
RMOD	par R
TRIE	par D
ACT	par C
CHOI	par H
INF	par NF
MD	par MD
SUP	par SP

et faites les changements de lignes, substitutions et suppressions suivants :

```
1100 REM S-P *CRETAB*
1110 DIM N$(50)
1120 DIM M$(50)
1130 DIM S$(50)
1140 DIM T$(50)
1150 DIM C$(50)
1160 DIM R$(50)
1170 DIM X$(50)
```

Effacez les lignes 1180-1190.

```
1210 LET Z=0
1220 LET R=0
1230 LET D=1
1240 LET C=0
1250 LET Z$="@PREMIER"
1260 LET Q$=""
1300 RETURN
```

Voici la version du sous-programme 1400 pour le Dragon 32 :

```
1400 REM S-P *LECFCR*
1410 OPEN "I",#-1,"ADBKDAT"
1420 PRINT "INSÉREZ LA CASSETTE DE
     DONNÉES ET APPUYEZ SUR "PLAY""
1430 GOSUB 3100
1440 INPUT#-1,Z
1450 FOR K=1 TO Z-1
1460 INPUT#-1, N$(K),M$(K),S$(K),T$(K),C$(K),
     R$(K),X$(K)
1470 NEXT K
1480 Q$=N$(1)
1490 CLOSE#-1
1500 PRINT "ARRÊTEZ LA CASSETTE"
1510 GOSUB 3100
1520 RETURN
```

Dans le précédent listage, pour le BBC Micro, remplacez la ligne 1410 par :

```
1410 F1=OPENIN("ADBKDAT")
```

Remplacez #-1par #F aux lignes 1440, 1460 et 1490.

Dans le précédent listage, pour le Commodore 64 et le Vic-20 remplacez la ligne 1410 par :

```
1410 OPEN 1,1,0,"ADBKDAT"
```

Remplacez #-1 par #1 aux lignes 1440 et 1460, et remplacez 1490 par :

```
1490 CLOSE 1
```

Sur le BBC Micro, remplacez INKEY$ par INKEY$(0) partout ; et remplacez INPUT"...";A$ par INPUT "...",A$. Sur les Commodore, remplacez LET A$=INKEY$ par GET A$ partout ; et remplacez IF INKEY$... par GET GET$: IF GT$..

Sur le BBC Micro, le Dragon et les Commodore, dans le sous-programme 4500 remplacez toutes les références à D$(L) par MID$(D$,L,1). Remplacez CODE... par ASC(...). Remplacez PREMIER par DERNIER à la ligne 4610.
Supprimez : LET L=T en fin de ligne 4630.

Voici la version du sous-programme 5600 pour le Dragon ; pour les versions BBC et Commodore, voir plus haut.

```
5600 REM S-P *SAUVENR*
5610 OPEN "O",#-1,"ADBKDAT"
5620 PRINT "INSÉREZ LA CASSETTE DE
     DONNÉES ET APPUYEZ SUR "RECORD""
5630 GOSUB 3100
5640 PRINT #-1,Z
5650 FOR K=1 TO Z-1
5660 PRINT #-1,Z,N$(K),M$(K),S$(K),T$(K),C$(K),
     R$(K),X$(K)
5670 NEXT K
5680 CLOSE #-1
5690 PRINT "ARRÊTEZ LA CASSETTE"
5693 GOSUB 3100
5695 RETURN
```

Dans le sous-programme 6200 pour le BBC Micro, insérez :

```
6205 VDU 2
6275 VDU 3
```

et remplacez LPRINT par PRINT.

Pour les Commodore, insérez :

```
6205 OPEN 4,4:CMD 4
6275 PRINT #4:CLOSE 4
```

et remplacez LPRINT par PRINT.

Pour le Dragon, remplacez LPRINT par PRINT#-2.

Les pères fondateurs

Dans l'histoire du micro-ordinateur, matériels et logiciels connaissent des progrès extrêmement liés. En ce domaine, les créateurs ont autant d'importance que leurs produits.

Au cours de l'histoire, les changements technologiques ont plus d'une fois bouleversé les habitudes. Mais rien n'égale la vitesse à laquelle s'est propagée la révolution micro-électronique, même pas le développement de l'aéronautique, de Louis Blériot à la navette spatiale. En moins de dix ans, on est passé des premiers microprocesseurs peu performants aux 16 bits d'aujourd'hui.

Aux alentours de 1971, des responsables de firmes californiennes qui fabriquaient des puces électroniques estimèrent possible de faire tenir un ordinateur entier sur une simple couche de silicium. A cette époque, personne n'envisageait de grandiose révolution, ni ne parlait de « technologie de l'information ». On pensait simplement à construire un petit ordinateur bon marché, qui pourrait contrôler des machines-outils, ou des ascenseurs : les premiers modèles furent d'ailleurs construits dans ce dessein.

On attribue généralement à la société Intel la mise au point du premier microprocesseur, le 4004. Il s'appelait ainsi parce que c'était un processeur de 4 bits, qui manipulait les données par blocs de quatre chiffres binaires. Sa mémoire était très limitée — mais bien suffisante pour un programme de contrôle pour ascenseur.

Dès 1972, Intel avait mené à bien la réalisation du 8008, un 8 bits, et de nombreux fervents d'électronique songèrent à construire autour un ordinateur personnel. Des articles parus dans la presse spécialisée américaine expliquèrent comment procéder; sans écran ni véritable clavier, ces appareils furent quand même les premiers ordinateurs domestiques. Le premier modèle commercial, l'Altair 8800, vient de là. Mais, en fait, beaucoup de spécialistes considèrent que le véritable précurseur des micro-ordinateurs est le Micral français, construit en 1972.

L'année suivante vit naître le 8080, toujours dû à Intel, dans lequel il faut voir le premier « véritable » microprocesseur. Il manipulait des données par blocs de 8 bits et pouvait disposer d'une mémoire allant jusqu'à 64 K. Les autres firmes entreprirent de répliquer. Le 6800 de Motorola était très semblable, mais il obéissait à un jeu d'instructions différent. De là naquirent les inextricables problèmes de compatibilité entre logiciels : les programmes écrits pour le 8080 ne pouvaient tourner sur le 6800, et inversement.

D'autres compagnies, parmi lesquelles on peut citer National Semiconductor, Signetics et Advanced Micro Devices, s'étaient elles aussi lancées dans la production de puces. Mais le grand innovateur en ce domaine fut MOS Technology, où travaillait Chuck Peddle. Cette firme mit au point le 6500, tellement semblable au 6800 de Motorola qu'il fallut lui faire subir de nombreuses modifications; on lui donna pour finir le nom de 6502.

Commodore s'était déjà fait un nom au Canada pour son matériel de bureau et ses calculatrices électroniques. Peddle y vint avec l'idée de construire, autour du 6502, un ordinateur personnel complet, avec écran, clavier, lecteur de cassettes pour servir de mémoire auxiliaire — bref, tout ce qu'il fallait. Ce fut le PET 2001, sorti en 1976, et dont le nom même (*pet* signifie « favori », « préféré ») entendait montrer que l'ordinateur pouvait s'adresser aussi au simple particulier.

Pères fondateurs
Bien que Chuck Peddle ait créé à la fois le microprocesseur 6502 et le PET Commodore, premier ordinateur familial, la contribution de Bill Gates n'est pas moins importante : c'est lui qui écrivit la version Microsoft du BASIC implanté dans la ROM de l'appareil. (Cl. Tony Sleep/Commodore.)

Chuck Peddle

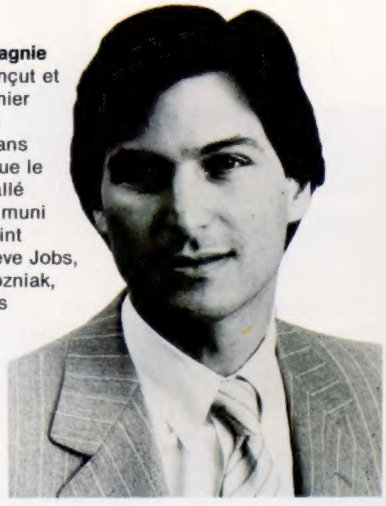
Bill Gates

Mais, au moment même où le PET était mis en vente, deux autres innovateurs s'affairaient au fond d'un garage californien. Steve Wozniak avait toujours rêvé d'avoir un ordinateur; la fréquentation d'un groupe d'amateurs, le Homebrew Computer Club, lui montra que c'était possible. Son premier modèle était construit à partir d'un simple circuit imprimé. Avec son ami Steve Jobs, il en fabriqua un certain nombre qu'il vendit. C'était l'Apple I (un boîtier muni d'un clavier), le père de l'Apple II, dont le succès allait être phénoménal. Apparu juste après le PET, l'appareil devait susciter la

création de multiples entreprises vouées à la fabrication de matériels et de logiciels.

La firme Tandy, installée à Fort Worth (Texas), avait elle aussi des idées sur le sujet. Elle était déjà — elle est toujours — un gros producteur de chaînes haute fidélité, de postes de radio et de synthétiseurs : passer aux ordinateurs domestiques ne représentait pour elle qu'une simple extension du marché, en profitant de surcroît de sa chaîne de distribution nationale, les boutiques Radio Shack. C'est d'ailleurs pourquoi le modèle qu'elle mit en vente (ce fut aussi un énorme succès) s'appelait TRS-80 Model I : TRS signifie *Tandy Radio Shack,* et 80 fait référence au matériel utilisé, un Z80 de chez Zilog. Cette dernière firme avait mis au point un microprocesseur assez semblable au 8080 d'Intel, mais en lui apportant de nombreuses améliorations.

Le PET et l'Apple II étaient donc bâtis autour d'un 6502, tandis que le TRS-80 recourait aux services d'un Z80. Le marché de la micro-informatique se diversifiait déjà. Mais ce problème se doublait d'un autre : une puce donnée détermine un certain éventail de logiciels, d'où des incompatibilités inévitables. Avec l'établissement de standards concernant le matériel, le même phénomène se produisit au niveau des programmes.



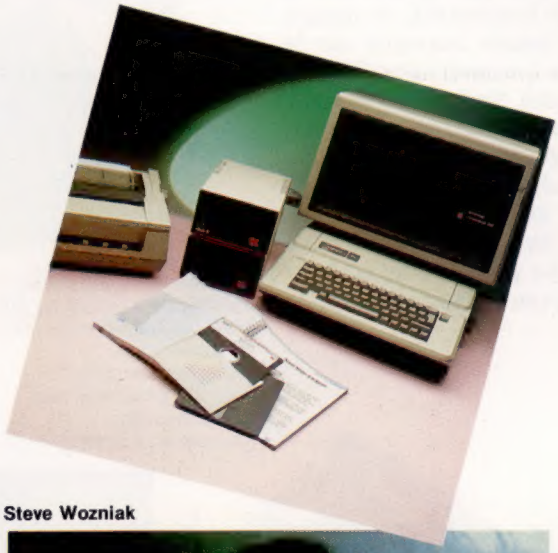
L'âme de la compagnie
Steve Wozniak conçut et construisit le premier Apple I (un simple circuit imprimé) dans son garage. Lorsque le dispositif fut installé dans un boîtier et muni d'un clavier, il devint l'Apple II, dont Steve Jobs, un vieil ami de Wozniak, fit l'énorme succès commercial qu'il est aujourd'hui. (Cl. Apple.)

Steve Jobs

En 1972, un jeune homme nommé Gary Kildall travaillait chez Intel comme consultant. Sa compagnie, Microprocessor Application Associates, mettait au point un langage de programmation qui devait permettre aux ingénieurs d'Intel de rédiger des programmes destinés aux puces produites par la firme. L'idée de Kildall était de relier un microprocesseur à un lecteur de disquettes 8 pouces et à un télétype, afin que chaque scientifique dispose de son propre ordinateur. Mais Intel préférait s'en tenir à l'utilisation d'un gros ordinateur en temps partagé.

Kildall et un de ses amis nommé John Torode construisirent eux-mêmes le dispositif, encore dans un garage californien. Torode réalisa la liaison entre le microprocesseur et le lecteur de disquettes, et Kildall rédigea le programme qui permettait au premier de contrôler le second, l'appelant CP/M *(Control Program/Microcomputers),* par allusion au langage de programmation (le PL/M ou *Programming Langage/Microcomputers*) qu'il avait mis au point chez Intel.

Ce premier système d'exploitation à être installé sur disquette fut rapidement repris par les constructeurs qui désiraient doter leurs appareils de ce périphérique. Il eut aussi des conséquences concernant les programmes : il ne tournait que sur le 8080 et sur le 8085 (plus rapide) d'Intel, ainsi que sur le Z80 de Zilog. Le Z80 devint donc un standard pour toute machine fonctionnant sous CP/M. Assurer la compatibilité avec celui-ci devint l'un des premiers objectifs des firmes de logiciels.

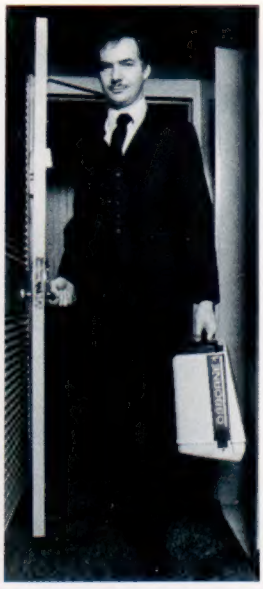


Steve Wozniak

Les ordinateurs domestiques avaient par ailleurs besoin d'un langage de programmation permettant aux particuliers d'écrire leurs propres programmes. Le BASIC, né à l'université de Dartmouth, aux États-Unis, était facile à apprendre et constituait de ce fait un choix tout désigné.

Bill Gates, un universitaire de Seattle, mit au point sur une puce un interpréteur BASIC, capable d'assurer la traduction des instructions en langage machine. Sa compagnie, Microsoft,

Gary Kildall
Les systèmes d'exploitation sont aujourd'hui mis au point par d'importantes équipes de programmeurs ; mais Kildall écrivit seul le CP/M. Le programme était prévu pour tourner sur du matériel encore très primitif, ce qui se remarquait encore par la suite.



Adam Osborne
A la fois aventurier et gestionnaire, Adam Osborne fut d'abord, plusieurs années durant, un des meilleurs journalistes spécialisés dans l'informatique, avant de fonder sa propre compagnie et de produire le premier ordinateur portable. (Cl. Osborne.)

devint la plus grande créatrice de langages de programmation, tout comme Digital Research régnait sur les systèmes d'exploitation. La fortune de Gates était faite.

De multiples avancées technologiques continuèrent de voir le jour. Dan Bricklin et Bob Frankston écrivirent Visicalc, le premier tableur (ou feuille de calcul), au sein de leur compagnie, Software Arts. Ce programme, distribué par Personal Software et tournant sur Apple II, devint le best-seller des programmes d'application, à tel point que Personal Software changea de raison sociale pour devenir VisiCorp. Un traitement de texte produit par MicroPro, la compagnie de Seymour Rubinstein, et appelé Word Star, connut un succès analogue sur le marché des programmes tournant sous CP/M.

Tous ces logiciels étaient destinés à des ordinateurs qui devenaient sans cesse plus puissants, et surtout moins onéreux. Un Britannique installé aux États-Unis, Adam Osborne, d'abord journaliste et rédacteur technique, puis éditeur de programmes, lança un ordinateur portable orienté vers la gestion. Le prix en était intéressant, et par surcroît l'appareil était vendu avec

Sir Clive Sinclair
Après bien des innovations dans de nombreux domaines, le succès sans précédent de ses micro-ordinateurs (ZX-80, ZX-81 et Spectrum) lui valut d'être fait chevalier par la reine d'Angleterre en 1983. (Cl. Sinclair Research.)

plusieurs logiciels intégrés. Il y eut aussi sir Clive Sinclair, dont les ZX-80, ZX-81 et Spectrum, par leur prix extrêmement bas, ont permis à des millions de débutants d'aborder l'informatique pour la première fois.

Le lancement par IBM, en 1982, de l'IBM PC a fait naître aussitôt un nouveau standard : tous les constructeurs et toutes les firmes productrices de logiciels ont entrepris très vite de travailler pour lui, ce qui n'a pu qu'inciter l'acheteur à se tourner vers cet appareil.

L'IBM PC a d'ailleurs fait appel à de nombreux pionniers de l'époque héroïque. Son microprocesseur vient de chez Intel; les systèmes d'exploitation sont dus à Microsoft, de Bill Gates, et à Digital Research, de Gary Kildall; et Visicalc et Wordstar furent parmi les premiers logiciels installés sur la machine.

Steve Wozniak et Steve Jobs sont toujours à la tête d'Apple, en concurrence directe avec IBM, et font reposer tous leurs espoirs sur la technologie révolutionnaire de Lisa et du tout récent Mac Intosh (une sorte de version simplifiée de Lisa, valant aux alentours de 20 000 francs). Chuck Peddle a monté sa propre compagnie, Sirius, et a réussi à s'emparer d'une bonne part du marché britannique avant l'arrivée d'IBM. Il est cependant victime actuellement de graves difficultés financières.

Mais il fera encore parler de lui un jour ou l'autre. La brève histoire de la micro-informatique montre que les précurseurs survivent, même quand les multinationales s'efforcent de dicter les règles du jeu.

Des débuts modestes
La technologie des micro-ordinateurs a davantage tiré profit des calculatrices programmables sophistiquées de la première génération des mini-ordinateurs. (Cl. Hewlett Packard.)

Herman Hauser

England, England
Chris Curry et Herman Hauser, créateurs et directeurs de la firme Acorn Computers, ont mis sur le marché des appareils comme l'Atom, le BBC et l'Electron qui représentent d'importantes percées technologiques. (Cl. Judy Goldhill.)

Chris Curry

Big Brother
IBM ne reconnut qu'en 1982 l'intérêt des micro-ordinateurs; sa décision d'en construire un eut aussitôt les effets prévus. Pratiquement tous les nouveaux appareils destinés à la gestion se veulent compatibles avec l'IBM PC, afin de bénéficier d'un énorme marché de logiciels. (Cl. Ian McKinnell.)

aBc INFORMATIQUE

Pour classer et présenter dans votre bibliothèque les fascicules de votre collection, des reliures mobiles pratiques et élégantes sont en vente chez tous les marchands de journaux.
Dans chaque reliure, vous trouverez, dans l'enveloppe qui contient les deux lames métalliques de la reliure, un décalque portant les numéros 1 à 8, qui vous permettra de marquer vous-même le dos de chaque volume.
Pour relier les 12 fascicules qui composent un volume, vous devrez en retirer les couvertures sans endommager les agrafes métalliques. Le numérotage de la reliure puis la mise en place des fascicules doivent être effectués selon les instructions ci-dessous.

1. **Disposez à plat la reliure. Enlevez le papier de protection du décalque. Positionnez le décalque en faisant coïncider l'écran qui entoure le numéro choisi avec l'écran situé sur le dos de la reliure.**

2. **Avec la pointe d'un stylo à bille, frottez régulièrement le numéro à transférer, en exerçant une certaine pression et en débordant légérement.**

3. **Enlevez doucement le support : le numéro est reporté sur la reliure. Posez dessus le papier de protection du décalque et frottez largement avec un objet poli ou arrondi, de manière à assurer la parfaite adhérence des caractères transférés.**

4. **Retournez la reliure mobile. Introduisez, d'un côté seulement, dans les encoches pratiquées dans l'épaisseur de la couverture, l'une des extrémités des deux lames d'acier livrées avec la reliure.**

5. **Vérifiez le bon ordre des fascicules, puis passez les deux lames d'acier dans les agrafes supérieures et inférieures.**

6. **Introduisez les extrémités libres des lames d'acier dans les encoches correspondantes de la reliure.**

Ce système original, sans mécanisme visible, donne au volume l'apparence d'une reliure classique.
Nous vous conseillons d'avoir toujours une reliure d'avance pour mieux protéger vos fascicules au fur et à mesure de leur parution.

Protégés par une élégante reliure, vos numéros d'aBc Informatique seront plus faciles à consulter

Pour classer, répertorier, protéger vos fascicules d'ABC Informatique, les Éditions Atlas vous proposent des reliures élégantes, sobres, qui s'insèrent parfaitement dans votre bibliothèque.
Chacune contient 12 fascicules, les maintient, les préserve. Un système simple, résistant, vous permet de les assembler facilement. Elles sont en vente en permanence chez votre marchand de journaux.
Demandez-les !

Chaque reliure:

40 FF
295 FB
18 FS

Cod. N.M.P.P.: 6103

EDITIONS ATLAS